Guia do Usuário

McMaster

Desenvolvimento de Sistemas com Microcontroladores PIC

18/02/2011

Índice

1. Introdução 10
2. McMaster – Desenvolvimento de Sistemas com Microcontroladores PIC 12
2.1. Introdução 12
2.2. Visão Macro do Sistema 12
2.3. Módulos Padrão 13
2.3.1. Microcontrolador 13
2.3.2. LCD alfanumérico 13
2.3.3. Displays de leds com 7 segmentos 14
2.3.4. Leds 16
2.3.5. Teclado matricial 16
2.3.6. Buzzer 17
2.3.7. Memória E2PROM externa 18
2.3.8. Relógio de tempo real (RTC) 18
2.3.9. Comunicação serial RS-232 19
2.3.10. Conversão analógica / digital (A/D) 20
2.3.11. Jumpers 21
2.4. Periféricos Adicionais 21
2.4.1. Placa de experiências EXP01 22
2.4.2. Placa de experiências EXP02 23
2.4.3. Placa de experiências EXP03 23
2.4.4. Placa de experiências EXP04 24
2.4.5. Placa de experiências EXP05 24
2.4.6. Adaptador McMaster 25
2.4.7. Gravador ICD2BR 26
3. Experiência 1 - Leitura de uma tecla e acionamento de um led 28
3.1. Objetivo 28
3.2. Descrição 28
3.3. Esquema Elétrico 29

3.4. Fluxograma 30
3.5. Código 31
3.6. Dicas e Comentários 34
3.7. Exercícios Propostos 34
4. Experiência 2 – Contador Simplificado 35
4.1. Objetivo 35
4.2. Descrição 35
4.3. Esquema Elétrico 36
4.4. Fluxograma 37
4.5. Código 39
4.6. Dicas e Comentários 42
4.7. Exercícios Propostos 42
5. Experiência 3 – Pisca - Pisca 43
5.1. Objetivo 43
5.2. Descrição 43
5.3. Esquema Elétrico 44
5.4. Fluxograma 45
5.5. Código 47
5.6. Dicas e Comentários 50
5.7. Exercícios Propostos 50
6. Experiência 4 –Conversão BCD para displays de 7 segmentos 51
6.1. Objetivo 51
6.2. Descrição 51
6.3. Esquema Elétrico 53
6.4. Fluxograma 54
6.5. Código 56
6.6. Dicas e Comentários 60
6.7. Exercícios Propostos 60
7. Experiência 5 – Timer de segundos 61

7.1. Objetivo 61
7.2. Descrição 61
7.3. Esquema Elétrico 62
7.4. Fluxograma 63
7.5. Código 66
7.6. Dicas e Comentários 71
7.7. Exercícios Propostos 71
8. Experiência 6 – Aceso à memória de dados EEPROM 72
8.1. Objetivo 72
8.2. Descrição 72
8.3. Esquema Elétrico 73
8.4. Fluxograma 74
8.5. Código 77
8.6. Dicas e Comentários 82
8.7. Exercícios Propostos 82
9. Experiência 7 - Dimmer 83
9.1. Objetivo 83
9.2. Descrição 83
9.3. Esquema Elétrico 84
9.4. Fluxograma 85
9.5. Código 88
9.6. Dicas e Comentários 93
9.7. Exercícios Propostos 93
10. Experiência 8 – Botões, Leds e Buzzer 94
10.1. Objetivo 94
10.2. Descrição 94
10.3. Esquema Elétrico 95
10.4. Fluxograma 96
10.5. Código 100

10.6. Dicas e Comentários 105
10.7. Exercícios Propostos 105
11. Experiência 9 – Varredura de displays e utilização do TIMER 1 106
11.1. Objetivo 106
11.2. Descrição 106
11.3. Esquema Elétrico 108
11.4. Fluxograma 109
11.5. Código 112
11.6. Dicas e Comentários 127
11.7. Exercícios Propostos 127
12. Experiência 10 – Display de cristal líquido LCD 128
12.1. Objetivo 128
12.2. Descrição 128
12.3. Esquema Elétrico 129
12.4. Fluxograma 130
12.5. Código 135
12.6. Dicas e Comentários 142
12.7. Exercícios Propostos 142
13. Experiência 11 – Conversor A/D 143
13.1. Objetivo 143
13.2. Descrição 143
13.3. Esquema Elétrico 144
13.4. Fluxograma 145
13.5. Código 148
13.6. Dicas e Comentários 152
13.7. Exercícios Propostos 153
14. Experiência 12 – Conversão A/D via RC 154
14.1. Objetivo 154
14.2. Descrição 154

14.3. Esquema Elétrico 156
14.4. Fluxograma 157
14.5. Código 159
14.6. Dicas e Comentários 163
14.7. Exercícios Propostos 164
15. Experiência 13 – Leitura de jumpers via RC 165
15.1. Objetivo 165
15.2. Descrição 165
15.3. Esquema Elétrico 166
15.4. Fluxograma 167
15.5. Código 169
15.6. Dicas e Comentários 173
15.7. Exercícios Propostos 173
16. Experiência 14 – Modulo PWM 174
16.1. Objetivo 174
16.2. Descrição 174
16.3. Esquema Elétrico 176
16.4. Fluxograma 177
16.5. Código 180
16.6. Dicas e Comentários 187
16.7. Exercícios Propostos 187
17. Experiência 15 – Acesso às memórias de dados e programa 188
17.1. Objetivo 188
17.2. Descrição 188
17.3. Esquema Elétrico 189
17.4. Fluxograma 190
17.5. Código 195
17.6. Dicas e Comentários 204
17.7. Exercícios Propostos 204

18. Experiência 16 – Master I²C 205
18.1. Objetivo 205
18.2. Descrição 205
18.3. Esquema Elétrico 207
18.4. Fluxograma 208
18.5. Código 214
18.6. Dicas e Comentários 221
18.7. Exercícios Propostos 222
19. Experiência 17 – Comunicação serial RS232 via USART 223
19.1. Objetivo 223
19.2. Descrição 223
19.3. Esquema Elétrico 224
19.4. Fluxograma 225
19.5. Código 227
19.6. Dicas e Comentários 232
19.7. Exercícios Propostos 233
20. Experiência 18 – Teclado matricial 4x4 234
20.1. Objetivo 234
20.2. Descrição 234
20.3. Esquema Elétrico 235
20.4. Fluxograma 236
20.5. Código 240
20.6. Dicas e Comentários 246
20.7. Exercícios Propostos 246
21. Experiência 19 – Relógio de tempo real (RTC) 247
21.1. Objetivo 247
21.2. Descrição 247
21.3. Esquema Elétrico 248
21.4. Fluxograma 249

21.5. Código 253
21.6. Dicas e Comentários 258
21.7. Exercícios Propostos 258
22. Experiência 20 – Sistema de temperatura e tacômetro 259
22.1. Objetivo 259
22.2. Descrição 259
22.2.1. O sensor de temperatura 259
22.2.2. O aquecimento 259
22.2.3. O resfriamento 260
22.2.4. Comunicação serial 260
22.2.5. Considerações gerais 260
22.3. Esquema Elétrico 261
22.4. Fluxograma 262
22.5. Código 266
22.6. Dicas e Comentários 274
22.7. Exercícios Propostos 274
23. Exemplo placa de experiência EXP01 275
23.1. Objetivo 275
23.2. Descrição 275
24. Exemplo placa de experiência EXP02 276
24.1. Objetivo 276
24.2. Descrição 276
24.3. Esquema Elétrico 277
24.3.1. Placa principal 277
24.3.2. Placa de experiências 278
24.4. Código 279
25. Exemplo placa de experiência EXP03 283
25.1. Objetivo 283
25.2. Descrição 283

26. Esquema Elétrico 284
26.1. Placa principal 284
26.1.1. Placa de experiências 285
26.2. Código 287
27. Exemplo placa de experiência EXP04 298
27.1. Objetivo 298
27.2. Descrição 298
28. Exemplo placa de experiência EXP05 299
28.1. Objetivo 299
28.2. Descrição 299
28.3. Código 300
29. Software de comunicação serial 319
30. Software demo para teste do hardware 320
31. Apêndice A – Esquema elétrico completo do McMaster 321
32. Apêndice B – Esquema elétrico completo da placa de experiências EXP01 324
33. Apêndice B – Esquema elétrico completo da placa de experiências EXP02 325
34. Apêndice C – Esquema elétrico completo da placa de experiências EXP03 327
35. Apêndice D – Esquema elétrico completo da placa de experiências EXP04 329
36. Apêndice E – Esquema elétrico completo da placa de experiências EXP05 330
37. Apêndice D – Esquema elétrico completo da placa Adaptadora McMaster 336
37.1.1.1.1.1.1. Pinagem Adaptador McMaster 337
38. Certicado de Garantia 342

1. Introdução

Inicialmente gostaríamos de parabenizá-lo por estar adquirindo o sistema didático McMaster. Este sistema utiliza o microcontrolador PIC18F4550 como objeto central. Junto ao microcontrolador, uma série de periféricos foram adicionados. O objetivo é disponibilizar uma placa de desenvolvimento onde o usuário possa testar seus conhecimentos em software, sem se preocupar com a montagem do hardware. Basta escrever o software. Veja todos os recursos que o sistema oferece:

- LCD alfanumérico;
- Displays de leds de 7 segmentos;
- Teclado matricial;
- Leds;
- Buzzer;
- Memória serial EEPROM (protocolo I^2C);
- Relógio de tempo real (protocolo I^2C);
- Comunicação serial padrão RS232;
- Conversão A/D;
- Leitura de jumpers;
- Sensor de temperatura;
- Aquecedor;
- Ventilador;
- Tacômetro;
- Lâmpada Incandescente;
- Gravação in-circuit;
- Placas de experiências;
- Fonte de alimentação bivolt (interno ao módulo).
- Terminais de alimentação individuais 5VDC / 3A e 12VDC / 1A protegidas contra curto e sobre corrente

Aliado a todos estes recursos, utilizou-se o microcontrolador PIC18F4550 que é o mais completo da família 18F. Suas principais características são:

- Capacidade de processamento de 12MIPS;
- Fonte de clock interna de 31kHz e 8MHz do tipo RC;
- PLL interno para multiplicar a freqüência de clock;
- Prioridade no tratamento de interrupção é possível escolher entre alta ou baixa prioridade;
- Hardware de multiplicação 8X8 bits executado em 1 ciclo de máquina;
- Alta capacidade de corrente nos pinos de I/O, 25mA por pino;
- Três fontes de interrupção externa;
- Uma interrupção de mudança de estado, quatro fontes;
- Timer 0 de 8 ou 16 bits configurável;
- Timer 1 e 3 de 16 bits;

- Timer 2 de 8 bits;
- Módulo ECCP;
- Módulo MSSP (SPI e I2C);
- EUSART (com suporte ao barramento LIN);
- ADC de 10bits;
- HLVD;
- BOR;
- WDT.

Fazem parte do kit de desenvolvimento:

- 1 Sistema de Treinamento em Microcontroladores PIC McMaster; com entrada de alimentação bivolt;
- 1 PIC18F4550;
- 1 Manual do Usuário;
- 1 CD-ROM;
- 1 Placa de experiência EXP01
- 1 Placa de experiência EXP02
- 1 Placa de experiência EXP03
- 1 Placa de experiência EXP04
- 1 Placa de experiência EXP05
- 1 Cabo USB
- 1 Cabo Serial

2. McMaster – Desenvolvimento de Sistemas com Microcontroladores PIC

2.1. Introdução

O McMaster é um equipamento para desenvolvimento de sistemas completo para o estudo da tecnologia de microcontroladores Microchip e em particular ao estudo do microcontrolador PIC18F4550. Na verdade, este sistema serve para muito mais que simplesmente o aperfeiçoamento dos conhecimentos da família PIC. Com ele o usuário é capaz de criar projetos completos, colocando em teste também a eficiência de seus conceitos e algoritmos.

Tudo isso é possível porque este sistema foi desenvolvido pensando na didática de ensino e nos problemas mais comuns do mercado em relação ao uso de microcontroladores.

2.2. Visão Macro do Sistema

Nesta seção será abordado através de uma visão macro o conceito do sistema utilizado no McMaster.

Ele é composto de um gravador ICD2Br para o microcontrolador, o microcontrolador PIC central, os periféricos ligados ao microcontrolador (aos quais daremos o nome de periféricos padrões) e um conector de expansão para experiências onde novos periféricos (aos quais daremos o nome de periféricos adicionais ou placas de experiência) poderão ser ligados.

Todos os I/Os do microcontrolador estão disponíveis no conector de expansão para experiências. Com exceção dos pinos RB6 e RB7 que são utilizados pela gravação in-circuit, todos os outros I/Os estão ligados diretamente ao conector, ou seja, sem nenhum tipo de proteção. Apenas os pinos RB6 e RB7 foram isolados. Por este motivo, é muito importante que o usuário configure corretamente os I/Os do microcontrolador quando for utilizar o conector de expansão, pois neste caso, uma ligação errada pode danificar o microcontrolador. Se o usuário utilizar o conector de expansão apenas com placas oficiais de experiências a preocupação com a direção dos I/Os do microcontrolador não precisa ser tomada, uma vez que as placas de experiências e todo o McMaster foram projetadas a fim de evitar que uma configuração errada do microcontrolador coloque o sistema em risco. Portanto, mesmo que um pino do microcontrolador seja configurado como saída quando o correto seria entrada a integridade do sistema está garantida. É claro que este erro pode acarretar num mau funcionamento do sistema projetado, porém nunca existirá risco ao McMaster e às placas de experiências, desde que as mesmas sejam oficiais e/ou homologadas pelo fabricante.Com o intuito de maximizar a compatibilidade entre as famílias de microcontroladores suportadas pela McMaster com seus módulos padrão optou-se pela utilização de jumpers de configuração para que pontos importantes do circuito possam ser desabilitados. Desta forma, através de 7 jumpers, descritos de acordo com a seqüência em que estão dispostos na placa McMaster, podemos configurar a ferramenta de acordo com a aplicação desejada:

- Cap USB (RC3) – Este jumper seleciona a ligação do capacitor de desacoplamento da referência de tensão utilizada pelo módulo USB do PIC18F4550. Selecione “OFF” para que o capacitor não seja desligado pois a comunicação USB é utilizada e “ON” para que o capacitor seja desligado pois a

comunicação USB não é utilizada.**Obs**: quando for utilizado o PIC16F877A ou PIC18F4520 obrigatoriamente este jumper deverá permanecer na posição “ON”

- Data I^2C – Este jumper seleciona o pino de via de dado I^2C do microcontrolador utilizado com os periféricos padrões. Selecione “ON” quando utilizar a via de dado do PIC18F4550 e “OFF” quando utilizar a via de dado do PIC16F877A ou PIC18F4520.

  Obs: quando for utilizado o PIC18F4550 com a via de dado I^2C selecionada obrigatoriamente o jumper Coluna 1 (RB0) deverá permanecer na posição “OFF”.

- Clock I^2C – Este jumper seleciona o pino de via de clock I^2C do microcontrolador utilizado com os periféricos padrões. Selecione “ON” quando utilizar a via de dado do PIC18F4550 e “OFF” quando utilizar a via de dado do PIC16F877A ou PIC18F4520.

  Obs: quando for utilizado o PIC18F4550 com a via de clock I^2C selecionado obrigatoriamente o jumper Coluna 2 (RB1) deverá permanecer na posição “OFF”.

- Coluna 2 (RB1) – Este jumper desliga o pino RB1 utilizado para ler os botões da coluna 2 do teclado. Selecione “ON” quando utilizar a coluna 2 do teclado e “OFF” quando não utilizar a coluna 2 do teclado.
- Coluna 1 (RB0) – Este jumper desliga o pino RB0 utilizado para ler os botões da coluna 1 do teclado. Selecione “ON” quando utilizar a coluna 1 do teclado e “OFF” quando não utilizar a coluna 1 do teclado.
- Linha 1 / Display Milhar (RB4) – Este jumper desliga o pino RB4 utilizado para ativar a linha 1 do teclado matricial e o display do milhar. Selecione “ON” quando utilizar a linha 1 do teclado matricial e o display do milhar e “OFF” quando não utilizar a linha 1 do teclado matricial e o display do milhar.
- Leds Especiais (RC0, RC1 e RC2) – Este jumper desabilita os leds ligados aos pinos RC0, RC1 e RC2 utilizados pelos módulos CPP e TIMER1 do microcontrolador. Selecione “ON” quando utilizar os leds especiais e “OFF” quando não utilizar os leds especiais.

2.3. Módulos Padrão

Nesta seção serão abordados cada um dos módulos padrão do McMaster.

2.3.1. Microcontrolador

O sistema utiliza o microcontrolador PIC18F4550 como centro de todo o hardware. Este microcontrolador está ligado a todos os periféricos disponíveis, possibilitando o estudo de praticamente todas as suas funções. Devido também ao grande poder de recursos deste modelo de PIC, é possível, junto aos demais recursos da placa, o desenvolvimento de projetos simples e/ou complexos, como por exemplo um controlador de temperatura com algoritmo de controle PID.

2.3.2. LCD alfanumérico

Nos dias de hoje, qualquer programador sabe da importância da interface com o usuário dentro de um sistema qualquer. Por isso, é muito importante o aprendizado de operação de um display do tipo LCD. No caso do McMaster, este display possui 2 linhas de 16 caracteres cada, sendo um padrão de mercado atual. Possui um chip de controle próprio, com o qual é realizada a interface com o microcontrolador. Com este periférico os sistemas desenvolvidos no McMaster poderão possuir telas explicativas, informações claras e menus de navegação.

A comunicação com o LCD é paralela com 8 vias de dados. Além destas, mais duas vias são utilizadas para controlar o LCD, uma denominada de ENABLE e a outra de RS.

A comunicação com o LCD é somente de escrita, desta forma, o pino de R/W do LCD está diretamente ligado ao terra (GND), não permitindo a leitura do mesmo.

As 8 vias de dados do LCD estão ligadas ao PORTD do microcontrolador, de RD0 (LSB) até RD7 (MSB). O pino de ENABLE está conectado ao pino RE1 do PIC e o pino RS do LCD ao pino RE0 do microcontrolador.

Assim, o esquema de ligação segue a tabela abaixo:

PIC	**LCD**
RD0...RD7	D0...D7
RE0	RS
RE1	ENABLE
Terra (GND)	R/W

Para maiores informações a respeito do LCD pode-se consultar o data sheet contido no CD que acompanha o McMaster.

2.3.3. Displays de leds com 7 segmentos

O LCD é uma ótima ferramenta de informação ao usuário, porém, muitas vezes ele ainda é inviável devido a custo, capacidade de visualização a distância e iluminação ao ponto de muitos projetos ainda utilizarem os velhos e práticos displays de leds ainda que não sejam a melhor alternativa para interface. No McMaster optou-se pela utilização de displays de 7 segmentos, que são numéricos, mas que permitem a visualização de diversas letras através da combinação específica destes segmentos. Optou-se também por 4 dígitos, todos com os segmentos interligados e os controles (comum) independentes, possibilitando a operação por varredura.

Atualmente, é muito comum encontrar em produtos do mercado, a combinação de ambos os tipos de display, para uma visualização mais completa e eficiente. Com o McMaster esta combinação também é possível.

A conexão dos displays com o microcontrolador segue a tabela abaixo:

PIC	Segmento
RD0	A
RD1	B
RD2	C
RD3	D
RD4	E
RD5	F
RD6	G
RD7	DP

E as vias de seleção de cada um dos displays, a tabela seguir:

PIC	Display
RB4	Milhar
RB5	Centena
RB6	Dezena
RB7	Unidade

Para a ativação dos displays deve-se selecionar nível lógico 1 nos pinos de seleção. Os segmentos também são ativados com nível lógico 1.

2.3.4. Leds

O McMaster possui um grupo de 8 leds que compartilha o mesmo barramento que os displays de 7 segmentos e o display LCD.

Desta forma, o seu acionamento deve ser feito via varredura sendo que os leds estão conectados ao PORTD e a seleção é feita pelo pino RA4. Da mesma forma que os displays, os leds são ativados com nível lógico 1, tanto na via de seleção (RA4) como individualmente (PORTD).

2.3.5. Teclado matricial

A maioria dos sistemas desenvolvidos atualmente utilizam algum tipo de teclado para a entrada de dados pelo usuário. O McMaster está provido de um teclado matricial de 4 linhas e 4 colunas, totalizando 16 teclas. O acionamento das linhas do teclado é feito simultaneamente com os comuns dos displays de 7 segmentos. Desta forma, ao acionar o display da unidade aciona-se também a linha 4 do teclado matricial. Junto com o display da dezena a linha 3 e assim por diante.

A tabela a seguir mostra esta relação:

Pino PIC	Estado	Teclado Matricial	Display de 7 segmentos
RB7	1	linha 4 ativada	unidade ativada
	0	linha 4 desativada	unidade desativada
RB6	1	linha 3 ativada	dezena ativada
	0	linha 3 desativada	dezena desativada
RB5	1	linha 2 ativada	centena ativada
	0	linha 2 desativada	centena desativada
RB4	1	linha 1 ativada	milhar ativada
	0	linha 1 desativada	milhar desativada

As colunas podem ser lidas através dos pinos RB0, RB1, RB2 e RB3, conforme a tabela a seguir:

Pino PIC	Estado	Teclado Matricial
RB0	1	Alguma tecla pressionada na coluna 1
	0	Nenhuma tecla pressionada na coluna 1
RB1	1	Alguma tecla pressionada na coluna 2
	0	Nenhuma tecla pressionada na coluna 2
RB2	1	Alguma tecla pressionada na coluna 3
	0	Nenhuma tecla pressionada na coluna 3
RB3	1	Alguma tecla pressionada na coluna 4
	0	Nenhuma tecla pressionada na coluna 4

Vale observar que para o correto funcionamento do teclado os jumpers relacionados com os pinos RB0 e RB4 devem estar configurados na posição ON.

2.3.6. Buzzer

Para chamar a atenção do usuário e confirmar certas ações, cada vez mais os sistemas estão fazendo uso de técnicas sonoras, seja através de simples bips ou por complexas melodias. Para que os usuários não fiquem sem o uso deste recurso, disponibilizou-se também um buzzer piezoelétrico com oscilação comandada diretamente pelo PIC, tornando possível a criação de sons diversos.

O microcontrolador deve então gerar (através do software) uma onda quadrada capaz de excitar o buzzer. Para gerar um simples beep pode-se utilizar uma onda quadrada com freqüência em torno de 650Hz e duração aproximada de 100ms. O buzzer está conectado ao pino RE2 do microcontrolador.

2.3.7. Memória E2PROM externa

Além da memória E2PROM interna do PIC, o McMaster possui ainda uma memória externa do tipo serial, modelo 24LC256 com 32Kbytes disponíveis para uso. Esta memória está soquetada na placa, possibilitando a sua substituição por outros modelos compatíveis, com maior ou menor capacidade.

A comunicação com esta memória é do tipo I²C, estando diretamente ligada os pinos do PIC responsáveis por este padrão de comunicação.

PIC	Memória
RC3/RB1	Clock (SCL) – pino 6
RC4/RB0	Data (SDA) – pino 5

Novamente os jumpers de configurações relacionados devem estar habilitados para a utilização da memória serial externa. Note que a posição 3-2 habilita os pinos RC3 e RC4. Já a posição 2-1 habilita os pinos RB0 e RB1.

Como a memória serial compartilha o mesmo barramento I^2C do relógio de tempo real (PCF8583P) se faz necessária a utilização de endereços diferentes para que o barramento seja compatível com os dois periféricos. Sendo assim, escolheu-se o endereço 7h (111b) para a memória serial. Para maiores informações sobre o protocolo de comunicação da memória serial 24LC256 pode-se consultar o data sheet disponível no CD.

Para detalhes a respeito da configuração dos jumpers utilizados na comunicação I^2C consulte o Capítulo 2.

2.3.8. Relógio de tempo real (RTC)

Utilizando o mesmo barramento I^2C da memória serial, o McMaster possui um relógio de tempo real, modelo PCF8583P. Com este componente o usuário poderá criar sistemas que contenham informações como a hora e a data atual. O relógio utilizado é completo, ou seja, é capaz de contar dia, mês, ano (inclusive bissexto), semana, hora, minuto, segundo e milésimo de segundo. Além de poder ser configurado de formas diferentes. O data sheet deste componente está disponível no CD que acompanha o McMaster. Pelo mesmo motivo já comentado na memória serial, o relógio foi configurado para utilizar o endereço 0h (000b) a fim de poder compartilhar o mesmo barramento I^2C.

Como no caso da memória, os pinos responsáveis pela comunicação são:

PIC	Relógio RTC
RC3/RB1	Clock (SCL) – pino 6
RC4/RB0	Data (SDA) – pino 5

2.3.9. Comunicação serial RS-232

Quem não deseja que seu projeto se comunique com um computador atualmente? Esta é outra tendência de mercado que os profissionais não podem deixar de lado. Seja para a configuração de parâmetros, para a coleta de dados ou a visualização gráfica de informações, a interligação entre o kit e o computador é um recurso que não poderia ser deixado de lado.

Optou-se pela comunicação serial, padrão RS-232 através de um conector DB-9. A interface e ajuste de tensões necessárias a este padrão, em relação à operação do PIC (TTL) é feita por um CI dedicado. Internamente, as vias de TX e RX podem ser ligadas aos pinos da USART do PIC, possibilitando o uso deste recurso.

A ligação ao microcontrolador segue a tabela abaixo.

PIC	COM.
RC6	TX (saída)
RC7	RX (entrada)

Faz parte também do módulo de comunicação serial, o conector DB9 fêmea que segue a pinagem abaixo:

Pino	Função
1	-
2	TX (saída)
3	RX (entrada)
4	-
5	Terra (GND)
6	-
7	-
8	-
9	-

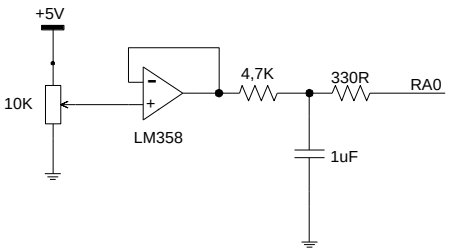
2.3.10. Conversão analógica / digital (A/D)

É verdade que estamos considerando o mundo cada vez mais digital, principalmente nos dias de hoje, onde vemos bilhões de informações trafegando por fibras ópticas e imagens de computador recriando o mundo real. Mas não podemos esquecer que a natureza é completamente analógica, e qualquer sistema que se baseie ou utilize informações deste meio externo precisará de um sistema de conversão para poder se comunicar. É por isso que, hoje e sempre, a conversão A/D é tão necessária.

Com o McMaster poderemos realizar estas conversões de duas maneiras. A primeira é através do conversor interno do PIC e a segunda é através de um pseudoconversor fundamentado no tempo de carga de um circuito RC.

Dentre os módulos padrão existem dois sistemas para trabalhar com o conversor A/D e para qualquer um deles, as duas formas de aquisição podem ser aplicadas, ou seja, tanto via A/D convencional como via RC. O primeiro sistema consiste num potenciômetro e o segundo num conjunto de jumpers que podem ser configurados como divisor resistivo ou circuito RC.Potenciômetro

O sistema com o potenciômetro segue o esquema elétrico representado a seguir.



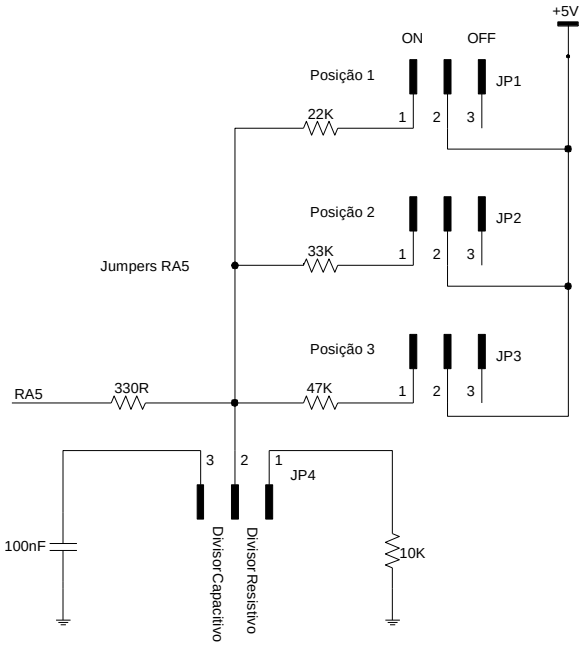
A tensão de entrada presente no pino RA0 do microcontrolador pode ser ajustada entre 0 e 5V. Caso se utilize o conversor A/D interno do PIC, o capacitor de 1uF e o resistor de 4K7 formam um filtro passa baixa útil para filtrar ruídos e deixar o sinal de entrada mais estável.

Caso se utilize o sistema de leitura via RC o conjunto de resistores e capacitores assume outra função. Neste sistema, para realizar a conversão deve-se executar as etapas a seguir:

- Inicialmente, através do software, deve-se descarregar o capacitor, colocando o pino do microcontrolador como saída em nível lógico 0. O capacitor se descarregará pelo resistor de 330R.
- Após o capacitor estar descarregado, coloca-se o pino do microcontrolador em entrada e começa-se a contar o tempo que o capacitor leva para se carregar (através do resistor de 4K7), ou seja, quanto tempo o capacitor leva para atingir nível lógico 1.
- Como tempo de carga é inversamente proporcional à tensão aplicada pelo potenciômetro, sabendo-se o tempo de carga pode-se estimar a tensão aplicada.

2.3.11. Jumpers

O sistema de jumpers está ligado ao pino RA5 do microcontrolador e segue o esquema elétrico representado a seguir.



Se configurarmos o sistema para divisor resistivo, basta ler com o conversor A/D do PIC a tensão presente no pino RA5 para estimar a posição do jumper.

Se configurarmos o sistema para resistor/capacitor, devemos seguir a mesma metodologia explicada no caso do potenciômetro, ou seja:

- Inicialmente descarregar o capacitor através do resistor de 330R colocando o pino do microcontrolador como saída em nível lógico 0.
- Após o capacitor estar descarregado, colocar o pino do microcontrolador em entrada e começar a contar o tempo que o capacitor leva para se carregar, ou seja, quanto tempo o capacitor leva para atingir nível lógico 1.
- Este tempo de carga é proporcional ao valor do circuito RC e portanto, pode ser utilizado para determinar a posição do jumper.

2.4. Periféricos Adicionais

A seguir serão explanados os periféricos adicionais (placas de expansão) contidos na placa de experiências EXP01, EXP02, EXP03, EXP04, EXP05 que acompanham o kit MCMASTER.

2.4.1. Placa de experiências EXP01

Entre outras funções a placa de experiências EXP01 possui um sistema completo para monitoramento e controle de temperatura, com um sensor e dois atuadores. Desta forma, tem-se um sensor de temperatura, um atuador de aquecimento (resistência controlada por PWM) e um atuador de resfriamento (ventilador controlado por PWM). Além disso, um sistema óptico ligado às hélices do ventilador é capaz de criar um tacógrafo, para monitoramento e controle de rotação.Possui também uma lâmpada incandescente além de gerar uma tensão de referência estável em 2,5V que pode ser utilizada como referência para o conversor A/D.Com tudo isso pode-se criar experimentos e projetos complexos de controle, começando em um simples controle ON/OFF até um avançado controlador PID.

2.4.1.1. Sensor de temperatura

A placa possui um circuito que utiliza um diodo de sinal como elemento sensor do medidor de temperatura ambiente. O sinal analógico proporcional à temperatura ambiente está presente no pino RA1 do microcontrolador e varia entre 0 e 5V.

Deve-se evitar que a temperatura ultrapasse 90ºC a fim de evitar que o sensor seja danificado.

2.4.1.2. Aquecedor

O aquecedor consiste numa resistência de 68Ω com 5W de dissipação. Pode ser acionada através do pino RC2 do microcontrolador. Veja que este pino pode ser configurado como PWM, e portanto, a potência de aquecimento pode ser regulada através deste recurso.

2.4.1.3. Ventilador

O sistema de ventilação consiste num cooler de PC que pode ser ativado através do pino RC1 do microcontrolador. Assim como no caso do aquecedor, este pino pode ser configurado como PWM, desta forma, pode-se modular a velocidade do ventilador utilizando este recurso do microcontrolador.

2.4.1.4. Tacômetro

Junto ao ventilador existe um sistema formado por um transmissor e um receptor de infravermelho. Este sistema é utilizado para medir a velocidade de rotação do ventilador. Quando não temos a passagem de luz, ou seja, quando a luz está interrompida por uma das palhetas do ventilador, o sistema de tacômetro apresentará na saída nível lógico 1. Quando se tem a passagem de luz, a saída do sistema de tacômetro será 0. O tacômetro está conectado ao pino RC0 (entrada de contador do TMR1) do microcontrolador.

2.4.1.5. Lâmpada incandescente

Consiste numa lâmpada incandescente de 12V que pode ser acionada através do pino RC5 do microcontrolador. Com nível lógico 1 a lâmpada acende e com nível lógico 0 a lâmpada apaga.

2.4.1.6. Tensão de referência

O circuito medidor de temperatura ambiente utiliza uma tensão de referência fixa e estável em 2,5V e como este recurso já estava presente na placa de experiências EXP01 resolveu-se também disponibilizar este recurso ao usuário. Assim, a tensão de referência de 2,5V foi conectada ao pino RA3 do PIC que pode ser configurado para utilizar este pino como entrada de referência externa do conversor A/D. Isto permite que o conversor A/D possa trabalhar em outra faixa de conversão e conseqüentemente com outra resolução.

2.4.2. Placa de experiências EXP02

Entre outras funções a placa de experiências EXP02 possui duas saídas analógicas ou PWM (configurável via jumper), duas saídas a relé (contatos NA ou NF), entrada de sinal isolada por fotoacoplador e saída isolada por fotoacoplador.

2.4.2.1. Conversor digital analógico - PWM

A placa possui um circuito que utiliza a saída do módulo CCP configurado como PWM e integra este sinal, de modo que gera uma saída DC proporcional ao ciclo ativo do PWM. Através de jumper de configuração podemos ter uma saída analógica ou PWM.

2.4.2.2. Saídas a RELÉ

A placa possui dois relés com contatos NA ou NF para ser utilizados em circuitos de potência como lâmpadas, motores, etc.

2.4.2.3. Entrada e saída de sinal isolada

A placa possui entrada e saída isolada para ligar sinais de outros sistemas onde há necessidade de isolação de sinais.

2.4.3. Placa de experiências EXP03

Entre outras funções a placa de experiências EXP03 possui dois potenciômetros digitais, jumpers para configuração ou forçar um determinado nível lógico (por exemplo, configuração de software, etc.) e entrada USB para experiências como RS232 emulada, mouse, etc.

2.4.3.1. Potênciometro digital

A placa possui um circuito que utiliza um potenciômetro digital de dois canais, controlados pela interface SPI, podendo substituir em boa parte dos casos o tradicional potenciômetro analógico.

2.4.3.2. Jumpers de configuração

A placa possui oito jumpers de configuração que podem ser utilizadas em diversos modos, como configuração de software, etc.

2.4.3.3. Entrada USB

A placa possui entrada USB para utilização com o PIC18F4550. Com esta entrada podemos emular, por exemplo, uma RS232, um mouse, entre outros.

2.4.4. Placa de experiências EXP04

A placa de experiências EXP04 é composta de uma matriz de contatos (proto-board) de 550 pontos, conector para derivação de sinais dos pinos do microcontrolador e conector com sinais de alimentação. A placa pode ser utilizada para montagem de circuitos experimentais complementando os já existentes nas placas de expansão e na placa McMaster.

2.4.5. Placa de experiências EXP05

A placa de experiências EXP05 é composta de inúmeros circuitos de entrada e saída para interfaceamento de sinais. Todas as conexões citadas estão disponibilizadas através de conectores tipo KRE.

2.4.5.1. Entradas e saídas optoacopladas

Para sinais de entrada, a placa dispõe de 8 entradas optoacopladas para sinais de 24VDC. Para sinais de saída, a placa dispõe de 8 saídas PNP optoacopladas com capacidade de corrente de 300mA. Os terras isolados de ambas interfaces, entrada e saída, bem como as entradas para as tensões de referência 24VDC estão interligados e disponibilizadas nos conectores CN8 e CN9 (referenciados como GND e VCC).

2.4.5.2. Entradas analógicas e saídas analógicas

Para sinais analógicos de entrada, a placa dispõe das seguintes interfaces: 1 entrada conversora analógico-digital para sinais de 0 a 10VDC, 1 entrada conversora analógico-digital para sinais de 4 a 20mA e 1 entrada conversora analógico-digital para amplificação de sinais (ganho igual 50). Para sinais analógicos de saída, a placa dispões de 2 saídas conversoras digital-analógico baseado em filtragem PWM. Os terras analógicos de ambas interfaces, entrada e saída, estão interligados ao terra do kit e disponibilizados no conector CN1 (referenciados como GND).

2.4.5.1. Interface de comunicação padrão RS-485

A placa também disponibiliza uma interface de comunicação RS-485, disponibilizada através dos terminais A e B do conector CN6, permitindo a conexão entre kits McMaster ou outros dispositivos compatíveis ao padrão. Como opcional o jumper de solda (JS1) permite incluir um resistor de terminação de linha de 120Ω.

2.4.6. Adaptador McMaster

A placa Adaptador McMaster pode ser substituída na posição ocupada pelo microcontrolador do McMaster permitindo a conexão de outros microcontraldores da família Microchip que sejam compatíveis ao esquema do Adaptador. Consulte o Capítulo 31 para mairores detalhes.

2.4.7. Gravador ICD2BR

Para utilizar o gravador presente no McMaster basta selecionar corretamente o seu modelo no software de desenvolvimento MPLAB da Microchip. Por se tratar de um gravador in-circuit o microcontrolador não precisa ser retirado da placa.

Ao habilitar o gravador no MPLAB o software atual do PIC18F4550 será paralisado e instantes após o final da gravação do novo software, o microcontrolador será automaticamente inicializado. Durante todas as

etapas de comunicação entre o ICD2BR e o microcontrolador (casos de gravação e depuração) o botão ICSP dever ser mantido pressionado (LED ICSP ON aceso).

Como se trata de um sistema de gravação conjugado ao McMaster, o usuário deverá sempre estabelecer a conexão USB entre o Kit e o MPLAB para correta operação do kit e execução dos exemplos.

Para a execução dos exemplos posteriormente gravados no microcontrolador, o usuário poderá decidir entre liberar a tecla ICSP (LED ICSP ON apagado) ou liberar via MPLAB o reset do ICD2BR;

Liberando o botão ICSP o microcontrolador sairá do estado de reset independentemente do comando de reset via MPLAB.

Obs: (1) O botão de reset do McMaster funcionará somente quando o reset do gravador (controlado através do software MPLAB) estiver liberado caso o botão ICSP esteja

pressionado (LED ICSP ON aceso)

(2) Para maiores detalhes a respeito da utilização do gravador ICD2BR consulte o seu manual de operação.

3. Experiência 1 - Leitura de uma tecla e acionamento de um led

3.1. Objetivo

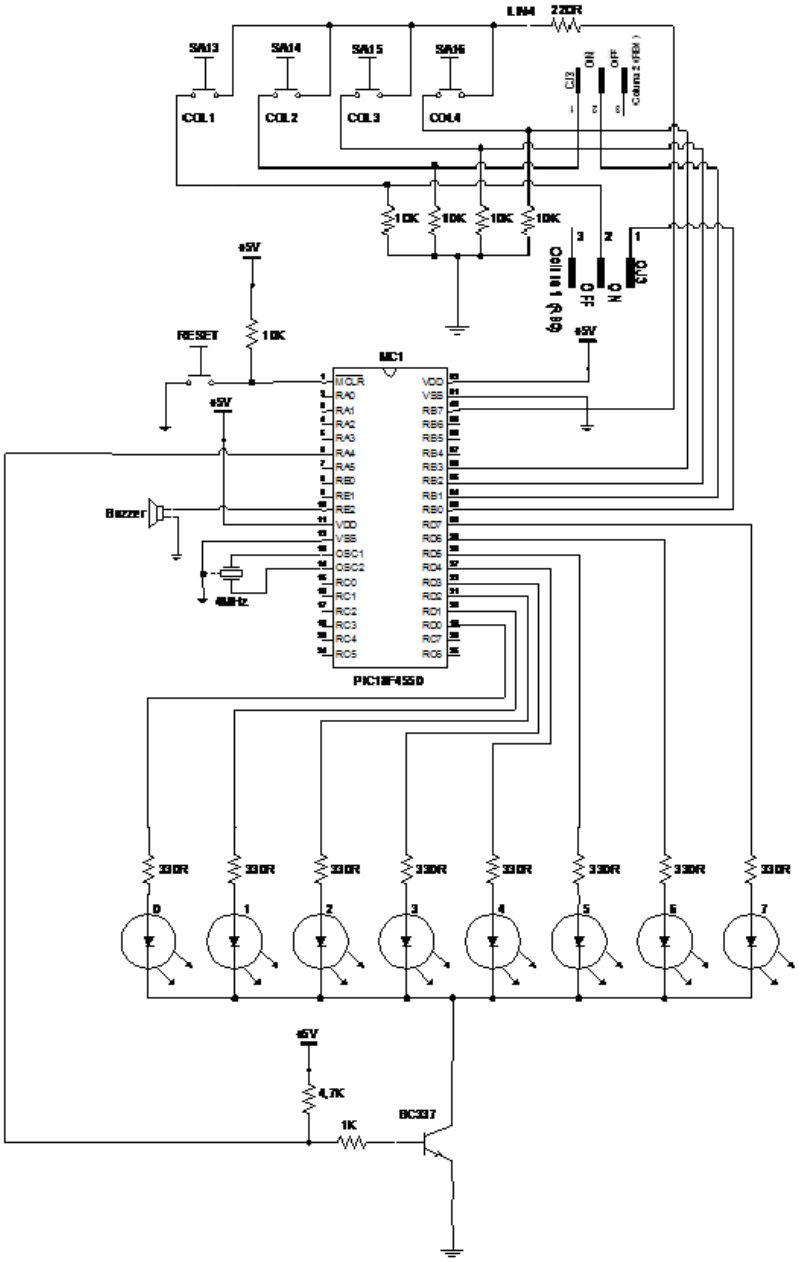
O objetivo desta experiência é ensinar ao aluno os primeiros passos sobre o microcontrolador. É apresentado o modo de configuração dos pinos de I/Os e as primeiras instruções utilizadas para testar condições nos pinos de entrada e alterações de estado nos pinos de saída, além de instruções para controle do fluxo do programa.

3.2. Descrição

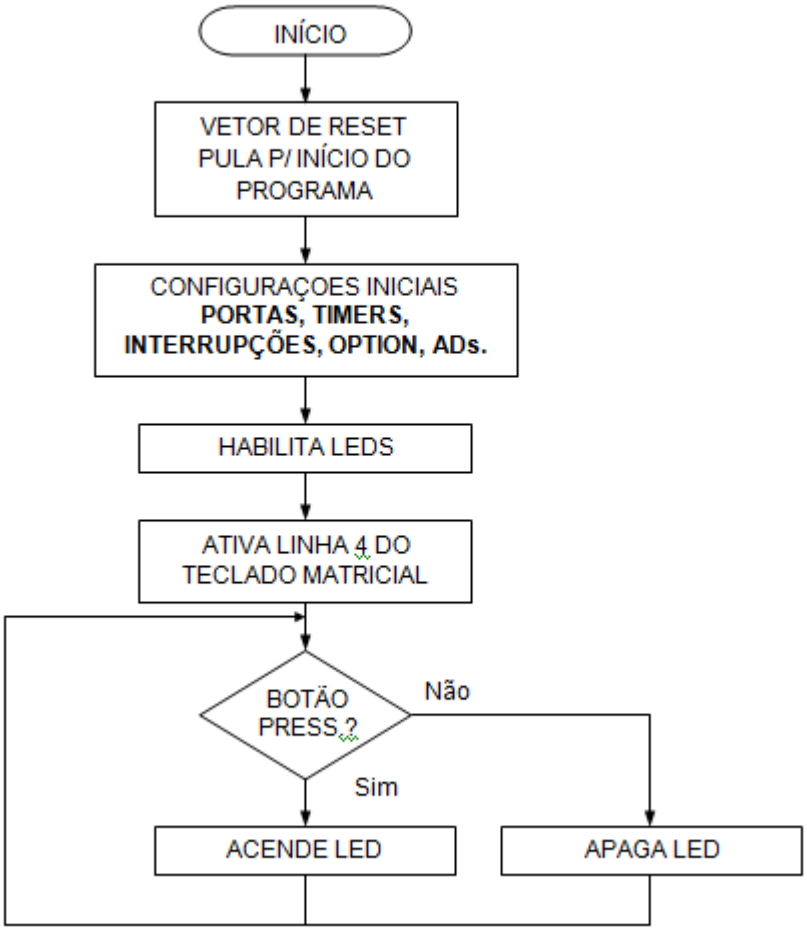
Sistema muito simples para representar o estado de um botão através de um led. Com o botão pressionado o led é ligado e com o botão solto o led é apagado.

O software inicia configurando os pinos de I/Os através dos registradores TRIS e dos registradores de periféricos pertinentes. Em seguida, o software habilita a linha 4 do teclado matricial e o grupo de leds ligados ao PORTD. A partir daí, o software entra num loop infinito onde o botão da linha 1 coluna 4 é testado e seu estado reproduzido no led 0 ligado ao pino RD0.

3.3. *Esquema Elétrico*



3.4. Fluxograma



3.5. Código

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Exemplo de Código para utilização com a McMaster            *
 *                                Exemplo 1                                *
 *                                                                         *
 *                                MOSAICO                                  *
 *                                                                         *
 *  TEL: (0XX11) 4992-8775                SITE: www.mosaico.com.br         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                          *
 *   DATA : 18/01/2010                                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            Descrição geral                              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

// Este software está preparado para demonstrar o controle dos pinos de I/O
// este primeiro programa demonstrará o estado de um botão por meio de um led.
/*
 CONFIGURAÇÂO DOS JUMPERS DA PLACA:

        CAP USB(RC3)                -> ON
        DATA I2C                    -> OFF
        CLOCK I2C                   -> OFF
        COLUNA 2(RB1)               -> ON
        COLUNA 1(RB0)               -> ON
        LINHA 1 / DISPLAY MILHAR(RB4)    -> ON
        LEDS ESPECIAS (RC0,RC1,RC2)      -> ON

*/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           DEFINIÇÃO PIC                                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <p18F4550.h>               // Register definitions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               INCLUDES DAS FUNÇÕES DE PERIFÉRICOS DO PIC                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <pwm.h>                //PWM library functions
#include <adc.h>                //ADC library functions
#include <timers.h>             //Timer library functions
#include <delays.h>             //Delay library functions
#include <i2c.h>                //I2C library functions
#include <stdlib.h>             //Library functions
#include <usart.h>              //USART library functions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Configurações para gravação                      *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#pragma config FOSC = HS
#pragma config CPUDIV = OSC1_PLL2
#pragma config WDT = OFF
#pragma config WDTPS = 128
#pragma config LVP = OFF
```

```
#pragma config PWRT = ON
#pragma config BOR = OFF
#pragma config BORV = 0
#pragma config PBADEN = OFF
#pragma config DEBUG = OFF
#pragma config PLLDIV = 1
#pragma config USBDIV = 1
#pragma config FCMEN = OFF
#pragma config IESO = OFF
#pragma config VREGEN = OFF
#pragma config MCLRE = ON
#pragma config LPT1OSC = OFF
#pragma config CCP2MX = ON
#pragma config STVREN = OFF
#pragma config ICPRT = OFF

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Definição e inicialização das variáveis Globais          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.
//Este programa não utiliza nenhuma variável de usuário

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Constantes internas                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de constantes facilita a programação e a manutenção.
//Este programa não utiliza nenhuma constante de usuário

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                    Declaração dos flags de software                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de flags ajuda na programação e economiza memória RAM.
//Este programa não utiliza nenhum flag de usuário

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Definição e inicialização dos port's                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       PROTOTIPAGEM DE FUNÇÕES                         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                                ENTRADAS                                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// As entradas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define BOTAO PORTBbits.RB0     //PORTA DO BOTÃO
                                //1 -> PRESSIONADO
                                //0 -> LIBERADO

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                                SAÍDAS                                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//As saídas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define    LED      PORTDbits.RD0   //PORTA DO LED
                                    //0 -> APAGADO
                                    //1 -> ACESO

#define    C_LEDS   PORTAbits.RA4   //PINO PARA ATIVAR GRUPO DE 8 LEDS
                                    //1 -> LEDS ATIVADOS
                                    //0 -> LEDS DESATIVADOS
```

```
#define    LINHA_4  PORTBbits.RB7   //PINO PARA ATIVAR LINHA 4 DO TECLADO
                                    //MATRICIAL
                                    //1 -> LINHA ATIVADA
                                    //0 -> LINHA DESATIVADA

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função Principal                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void main ()
{
    PORTA = 0x00;               //Limpa PORTA
    PORTB = 0x00;               //Limpa PORTB
    PORTC = 0x00;               //Limpa PORTC
    PORTD = 0x00;               //Limpa PORTD
    PORTE = 0x00;               //Limpa PORTE

    LATA = 0x00;                //Limpa PORTA
    LATB = 0x00;                //Limpa PORTB
    LATC = 0x00;                //Limpa PORTC
    LATD = 0x00;                //Limpa PORTD
    LATE = 0x00;                //Limpa PORTE

    TRISA = 0b00101111;         //CONFIG DIREÇÃO DOS PINOS PORTA
    TRISB = 0b00001111;         //CONFIG DIREÇÃO DOS PINOS PORTB
    TRISC = 0b10011001;         //CONFIG DIREÇÃO DOS PINOS PORTC
    TRISD = 0b00000000;         //CONFIG DIREÇÃO DOS PINOS PORTD
    TRISE = 0b00000000;         //CONFIG DIREÇÃO DOS PINOS PORTE

    ADCON1 = 0b00001111;        //DESLIGA CONVERSORES A/D

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Inicialização do Sistema                      *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
    C_LEDS = 1;                 //ATIVA LEDS LIGADOS AO PORTD

    LINHA_4 = 1;                //ATIVA LINHA 4 DO TECLADO MATRICIAL

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Rotina Principal                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
    while(1)
    {
        ClrWdt();
        if (BOTAO) LED = 1;     // testa botão. Se botão = 0, então led = 1
        else LED = 0;           // caso contrário, led = 0
    }

}   // FIM DO PROGRAMA
```

3.6. Dicas e Comentários

Veja que os pinos do microcontrolador, tanto os de entrada como os de saída são declarados através de DEFINES no início do software o que facilita futuras alterações na pinagem do hardware.

Repare também que o exemplo é extremamente simples e nenhum tipo de tratamento de debounce para a tecla foi utilizado.

3.7. Exercícios Propostos

1. Altere a lógica do sistema, ou seja, com o botão pressionado o led deve permanecer apagado e com o botão liberado o led deve permanecer acesso.
2. Altere o software a fim de trocar a tecla ativa, passando por exemplo a utilizar a tecla da linha 4 coluna 2.
3. Altere o software para ligar/desligar outro led, por exemplo, o led ligado ao pino RD3.

4. Experiência 2 – Contador Simplificado

4.1. Objetivo

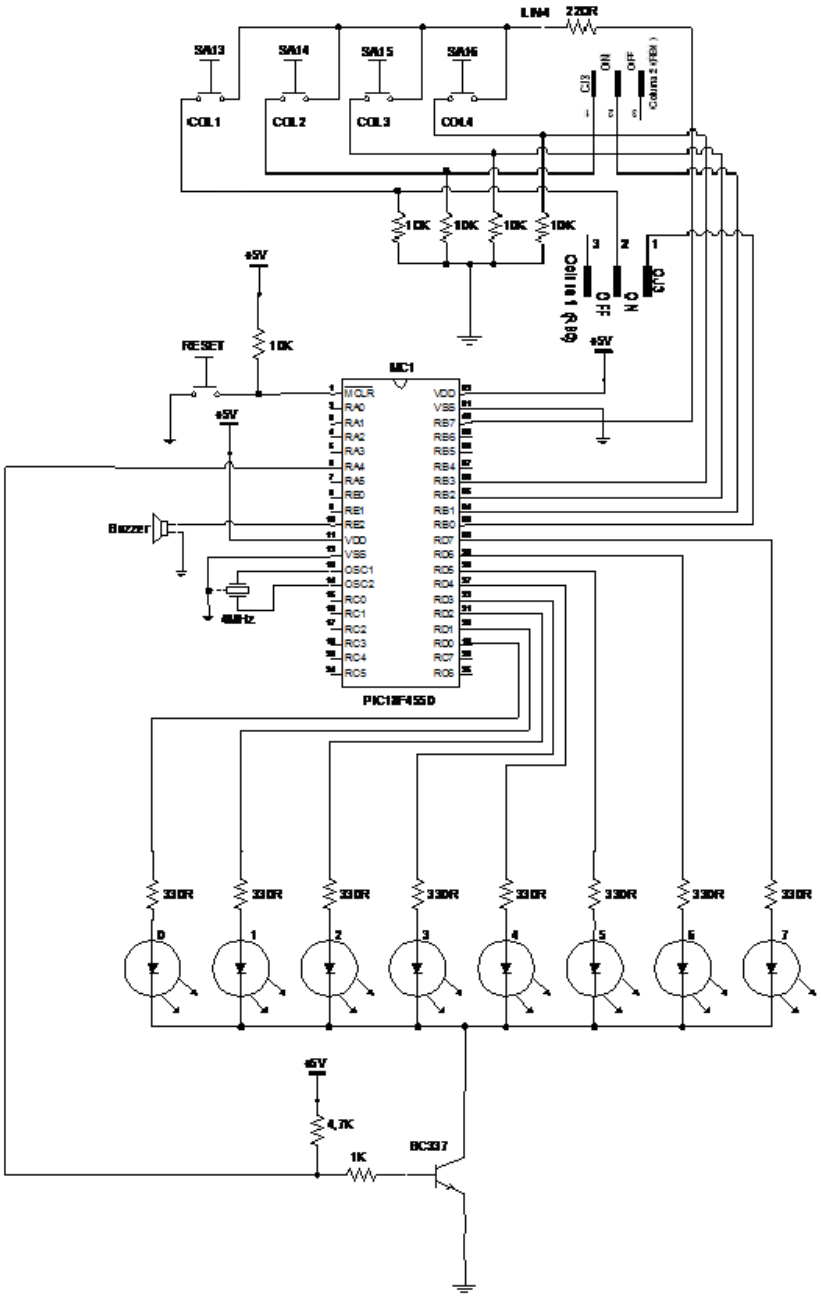
O objetivo desta experiência é ensinar os recursos de software comumente utilizados para tratamento de debounce de teclas e a manipulação de variáveis declaradas na RAM do microcontrolador.

4.2. Descrição

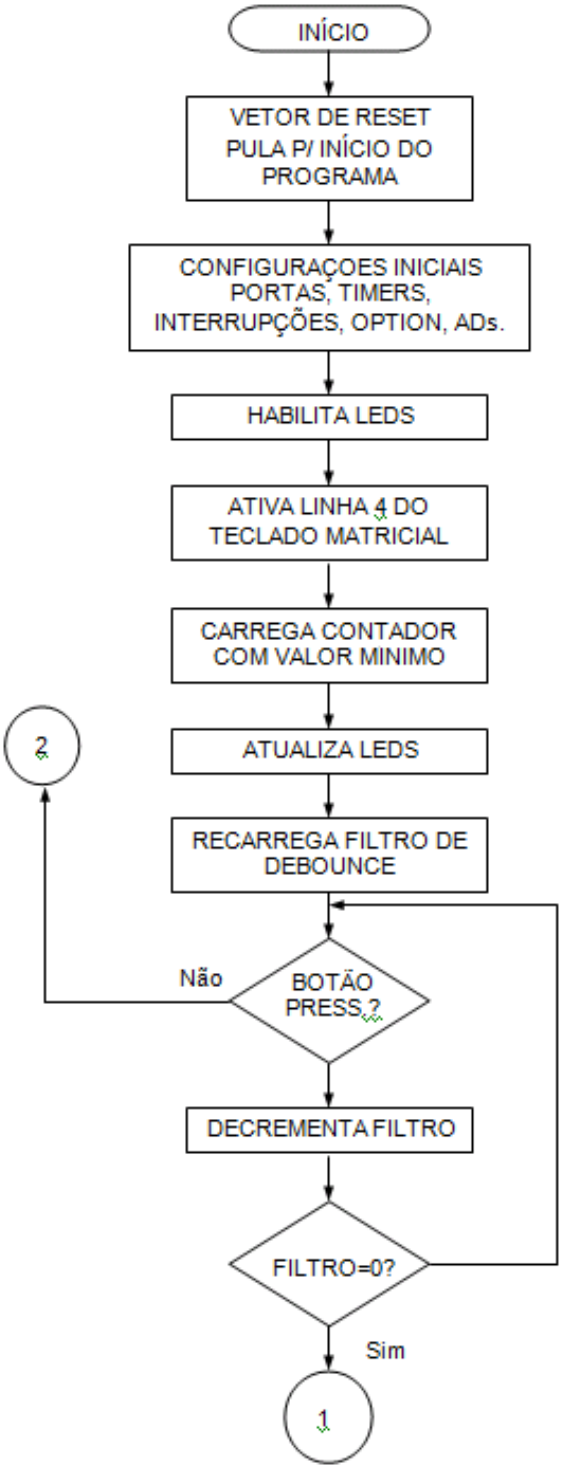
O software faz uso do grupo de leds para representar de forma binária o valor da variável “CONTADOR” declarada na RAM do microcontrolador. Utilizando o botão da linha 4 coluna 1 altera-se o valor da variável através de instruções de incremento e decremento. O valor está limitado por constantes declaradas no início do código. Como apenas um botão é utilizado, a variável é incrementada até o valor máximo e em seguida decrementada até o valor mínimo, permanecendo neste looping indefinidamente. Foi utilizado um flag para alterar o sentido da contagem sempre que um dos extremos é atingido.

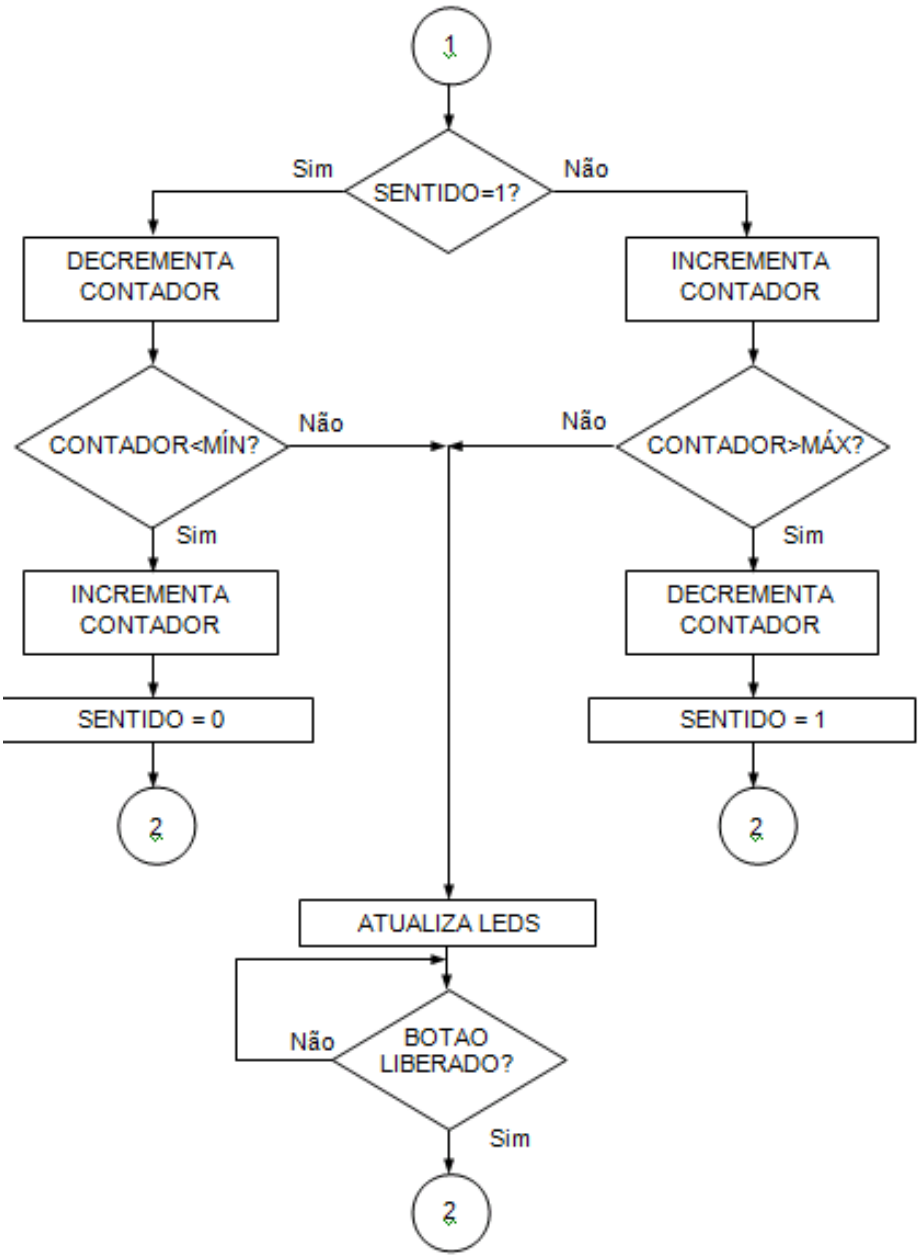
Foi feito o tratamento de debounce da tecla que consiste em testar repetidas vezes se a tecla foi realmente pressionada para somente depois executar a ação correspondente. Sempre que a tecla estiver solta o contador de debounce (variável “FILTRO”) é inicializado e sempre que a tecla for pressionada o valor da variável “FILTRO” é decrementado, de forma que a tecla somente é considerada pressionada quando o valor de “FILTRO” for igual a zero.

4.3. Esquema Elétrico



4.4. Fluxograma




1
SENTIDO=1?
Sim
Não
DECREMENTA CONTADOR
INCREMENTA CONTADOR
CONTADOR<MÍN?
CONTADOR>MÁX?
Não
Não
Sim
Sim
INCREMENTA CONTADOR
DECREMENTA CONTADOR
SENTIDO = 0
SENTIDO = 1
2
2
ATUALIZA LEDS
BOTAO LIBERADO?
Não
Sim
2

4.5. Código

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Exemplo de Código para utilização com a McMaster           *
 *                              Exemplo 2                                  *
 *                                                                         *
 *                               MOSAICO                                   *
 *                                                                         *
 *  TEL: (0XX11) 4992-8775                SITE: www.mosaico.com.br         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                          *
 *   DATA : 18/01/2010                                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Descrição geral                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

//Este exemplo mostra um contador simplificado

/*
 CONFIGURAÇÂO DOS JUMPERS DA PLACA:

        CAP USB(RC3)                -> ON
        DATA I2C                    -> OFF
        CLOCK I2C                   -> OFF
        COLUNA 2(RB1)               -> ON
        COLUNA 1(RB0)               -> ON
        LINHA 1 / DISPLAY MILHAR(RB4)    -> ON
        LEDS ESPECIAS (RC0,RC1,RC2)      -> ON

*/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            DEFINIÇÃO PIC                                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <p18F4550.h>               // Register definitions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               INCLUDES DAS FUNÇÕES DE PERIFÉRICOS DO PIC               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <pwm.h>                //PWM library functions
#include <adc.h>                //ADC library functions
#include <timers.h>             //Timer library functions
#include <delays.h>             //Delay library functions
#include <i2c.h>                //I2C library functions
#include <stdlib.h>             //Library functions
#include <usart.h>              //USART library functions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Configurações para gravação                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#pragma config FOSC = HS
#pragma config CPUDIV = OSC1_PLL2
#pragma config WDT = OFF
#pragma config WDTPS = 128
#pragma config LVP = OFF
#pragma config PWRT = ON
#pragma config BOR = OFF
#pragma config BORV = 0
#pragma config PBADEN = OFF
#pragma config DEBUG = OFF
#pragma config PLLDIV = 1
```

```
#pragma config USBDIV = 1
#pragma config FCMEN = OFF
#pragma config IESO = OFF
#pragma config VREGEN = OFF
#pragma config MCLRE = ON
#pragma config LPT1OSC = OFF
#pragma config CCP2MX = ON
#pragma config STVREN = OFF
#pragma config ICPRT = OFF

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Definição e inicialização das variáveis Globais          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.
struct flags
{
    unsigned char BIT0 :1;
    unsigned char BIT1 :1;
    unsigned char BIT2 :1;
    unsigned char BIT3 :1;
    unsigned char BIT4 :1;
    unsigned char BIT5 :1;
    unsigned char BIT6 :1;
    unsigned char BIT7 :1;
};

unsigned char    CONTADOR;
unsigned char    FILTRO;

struct flags FLAG_0;

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Constantes internas                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de constantes facilita a programação e a manutenção.

#define     MIN     10
#define     MAX     30

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Declaração dos flags de software                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de flags ajuda na programação e economiza memória RAM.

#define     sentido     FLAG_0.BIT0
#define     C_LEDS       LATAbits.LATA4
#define     LINHA_4     LATBbits.LATB4


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Definição e inicialização dos port's                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       PROTOTIPAGEM DE FUNÇÕES                         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              ENTRADAS                                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// As entradas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define BOTAO PORTBbits.RB0     //PORTA DO BOTÃO
                                //1 -> PRESSIONADO
                                //0 -> LIBERADO
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              SAÍDAS                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//As saídas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função Principal                             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void main ()
{
    PORTA = 0x00;               //Limpa PORTA
    PORTB = 0x00;               //Limpa PORTB
    PORTC = 0x00;               //Limpa PORTC
    PORTD = 0x00;               //Limpa PORTD
    PORTE = 0x00;               //Limpa PORTE

    LATA = 0x00;                //Limpa PORTA
    LATB = 0x00;                //Limpa PORTB
    LATC = 0x00;                //Limpa PORTC
    LATD = 0x00;                //Limpa PORTD
    LATE = 0x00;                //Limpa PORTE

    TRISA = 0b00101111;         //CONFIG DIREÇÃO DOS PINOS PORTA
    TRISB = 0b00001111;         //CONFIG DIREÇÃO DOS PINOS PORTB
    TRISC = 0b10011001;         //CONFIG DIREÇÃO DOS PINOS PORTC
    TRISD = 0b00000000;         //CONFIG DIREÇÃO DOS PINOS PORTD
    TRISE = 0b00000000;         //CONFIG DIREÇÃO DOS PINOS PORTE

    ADCON1 = 0b00001111;        //DESLIGA CONVERSORES A/D

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      Inicialização do Sistema                         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

    C_LEDS = 1;                 //ATIVA LEDS LIGADOS AO PORTD

    LINHA_4 = 1;                //ATIVA LINHA 4 DO TECLADO MATRICIAL

    sentido = 0;

    CONTADOR = MIN;
    PORTD = CONTADOR;

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Rotina Principal                             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
    while(1)
    {
        ClrWdt();
        FILTRO = 200;
        if(!BOTAO) continue;
        while(BOTAO)
        {
            ClrWdt();
            FILTRO--;
            if(FILTRO) continue;
            else break;
        }

        if(sentido == 0)
        {
            CONTADOR++;
            if(CONTADOR > MAX)
```

```
            {
                CONTADOR--;
                sentido = 1;
                continue;
            }
        }
        else
        {
            CONTADOR--;
            if(CONTADOR < MIN)
            {
                CONTADOR++;
                sentido = 0;
                continue;
            }

        }

        PORTD = CONTADOR;
        while(BOTAO) ClrWdt();
    }

}   // FIM DO PROGRAMA
```

4.6. Dicas e Comentários

O tempo do debounce pode ser calculado pelo tempo de execução da varredura da tecla multiplicado pelo valor inicial da variável de filtro. No exemplo estudado o tempo de debounce é de 1,15ms e foi calculado seguindo a equação abaixo:

Tempo de Debounce = ciclo de máquina X número de ciclos de máquina na varredura da tecla X valor inicial da variável “FILTRO”

Tempo de Debounce = 1us x 5 x 230

Tempo de Debounce = 1150us = 1,15ms

Não existe uma regra para calcular o tempo ideal que deva ser utilizado no debounce de teclas. Isso irá depender do projeto, do tipo de tecla em utilização, do ambiente de trabalho onde o sistema irá operar, etc. Na prática, tempos longos da ordem de grandeza de dezenas de milisegundos podem ser necessários.

4.7. Exercícios Propostos

1. Alterar os limites da contagem.
2. Retirar o tratamento de debounce do software e verificar os efeitos práticos.

5. Experiência 3 – Pisca - Pisca

5.1. Objetivo

O objetivo desta experiência é ensinar ao aluno com criar rotinas de delays além de apresentar uma técnica simples utilizada para inverter o estado de um bit.

5.2. Descrição

O software desta experiência utiliza um dos displays de 7 segmentos para implementar um pisca-pisca sendo que a freqüência das piscadas é controlada através do uso de uma rotina de delay.

A rotina de delay é genérica é pode ser utilizada para gerar delays entre 1ms e 256ms. Na realidade, a rotina recebe um argumento passado pelo WORK para determinar o atraso que deve ser gerado, sempre em múltiplos de 1ms. Como o argumento é de 8 bits existem 256 possíveis delays, indo de 1ms até 256ms. Basta portanto, carregar o WORK com o delay desejado e chamar a rotina.

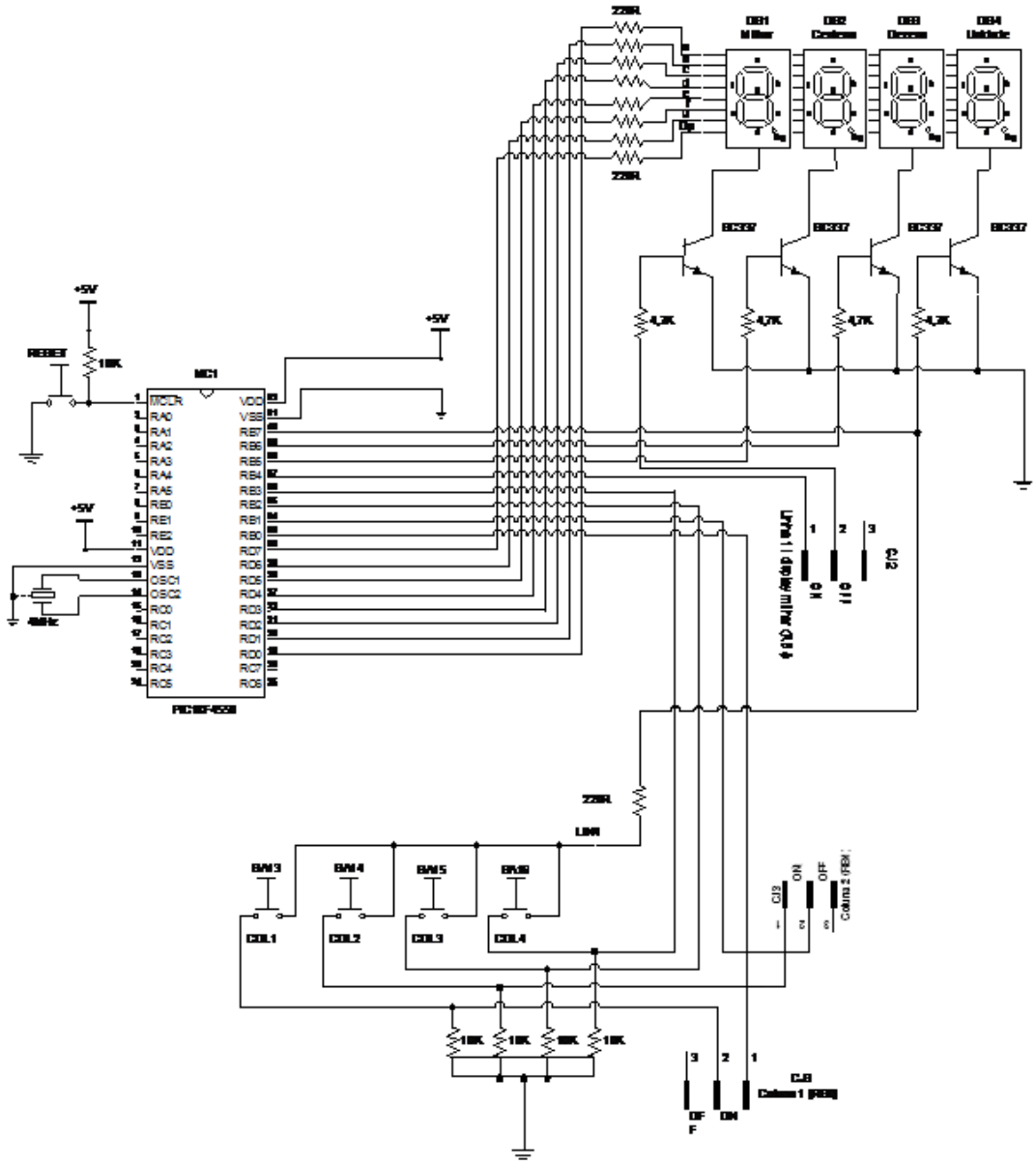
O pisca-pisca é visualizado no display de 7 segmentos na posição da unidade. Sempre que o delay é finalizado o PORTD deve ser invertido e para inverter o estado destes bits utilizou-se a operação lógica boleana XOR. Conforme a tabela verdade apresentada a seguir

A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

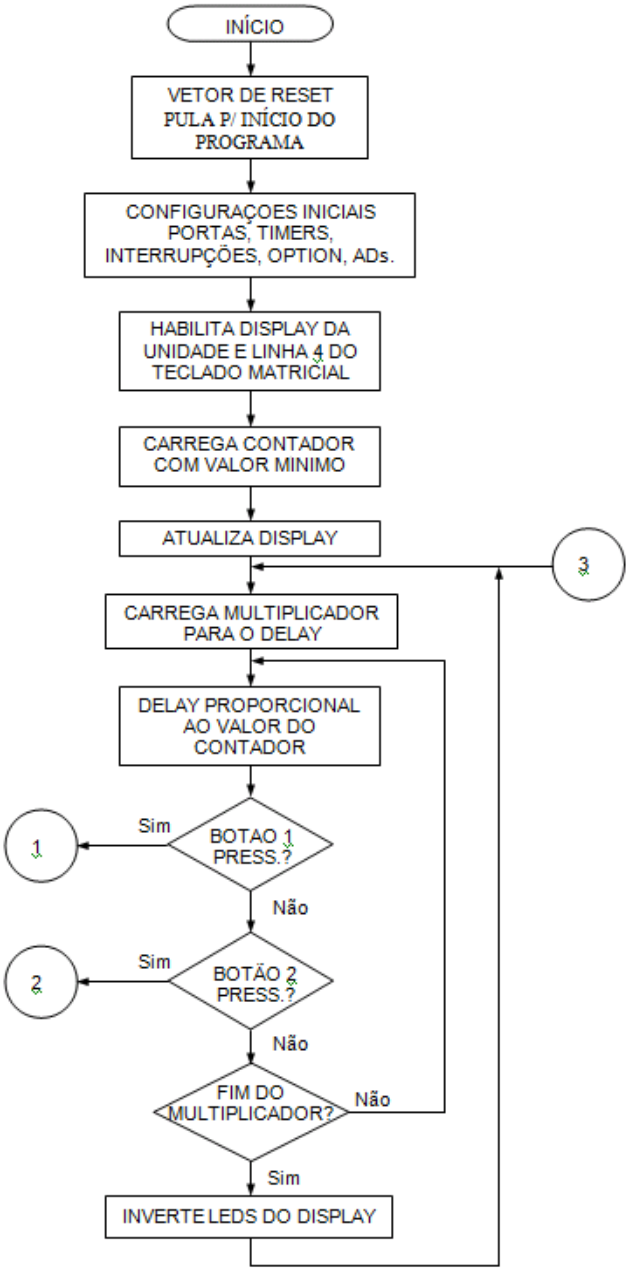
pode-se verificar que sempre que a operação é realizada quando os bits da coluna **A** estão em 1 o resultado fica invertido em relação à coluna **B** e sempre que os bits da coluna **A** estão em 0 o resultado se mantém em relação à coluna **B**. Assim, sempre que ser desejar inverter um bit, basta fazer uma operação XOR entre um bit em 1 e o bit que se deseja inverter. Esta é uma técnica simples de inverter o estado de um bit sem testar o estado original.

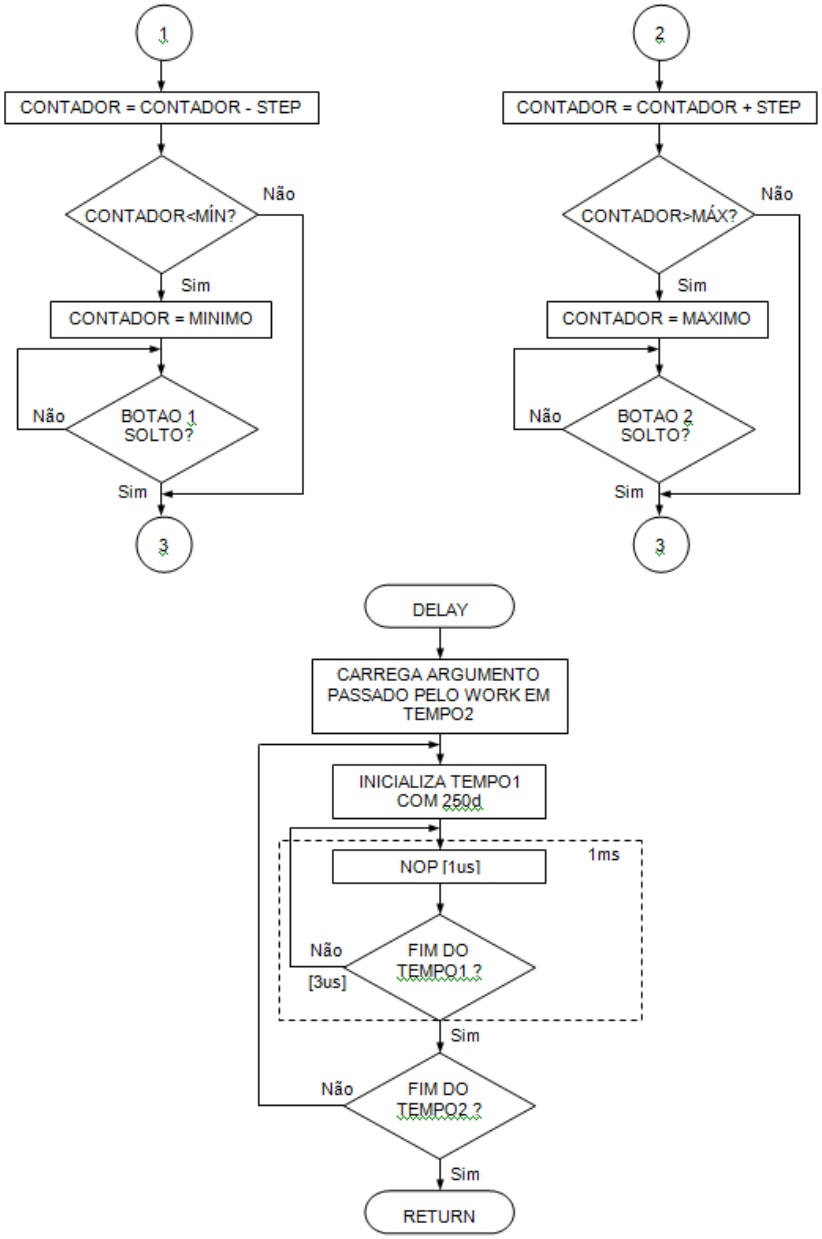
No software, pode-se utilizar as teclas da linha 4 colunas 1 e 2 para alterar o tempo do delay e conseqüentemente a freqüência das piscadas do display. A tecla da coluna 1 incrementa o valor do delay enquanto a tecla da coluna 2 decrementa.

5.3. Esquema Elétrico



5.4. Fluxograma




1
CONTADOR = CONTADOR - STEP
CONTADOR<MÍN?
Não
Sim
CONTADOR = MINIMO
Não
BOTAO 1 SOLTO?
Sim
3
2
CONTADOR = CONTADOR + STEP
CONTADOR>MÁX?
Não
Sim
CONTADOR = MAXIMO
Não
BOTAO 2 SOLTO?
Sim
3
DELAY
CARREGA ARGUMENTO PASSADO PELO WORK EM TEMPO2
INICIALIZA TEMPO1 COM 250d
1ms
NOP [1us]
Não
[3us]
FIM DO TEMPO1 ?
Sim
Não
FIM DO TEMPO2 ?
Sim
RETURN

5.5. Código

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Exemplo de Código para utilização com a McMaster            *
 *                              Exemplo 3                                  *
 *                                                                         *
 *                               MOSAICO                                   *
 *                                                                         *
 *  TEL: (0XX11) 4992-8775                SITE: www.mosaico.com.br         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                          *
 *   DATA : 18/01/2010                                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Descrição geral                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

//PISCA-PISCA VARIÁVEL PARA DEMONSTRAR A IMPLEMENTAÇÃO DE DELAYS E A INVERSÃO
//DE PORTAS. APENAS OS BOTÕES DA LINHA 4 ESTÃO ATIVADOS SENDO QUE O DA
//COLUNA 1 É UTILIZADO PARA INCREMENTAR O TEMPO ENTRE AS PISCADAS.
//O BOTÃO DA COLUNA 2 É UTILIZADO PARA DIMINUIR O TEMPO ENTRE AS PISCADAS.

/*
 CONFIGURAÇÂO DOS JUMPERS DA PLACA:

        CAP USB(RC3)              -> ON
        DATA I2C                  -> OFF
        CLOCK I2C                 -> OFF
        COLUNA 2(RB1)             -> ON
        COLUNA 1(RB0)             -> ON
        LINHA 1 / DISPLAY MILHAR(RB4)    -> ON
        LEDS ESPECIAS (RC0,RC1,RC2)      -> ON

*/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            DEFINIÇÃO PIC                                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <p18F4550.h>           // Register definitions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               INCLUDES DAS FUNÇÕES DE PERIFÉRICOS DO PIC                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <pwm.h>                //PWM library functions
#include <adc.h>                //ADC library functions
#include <timers.h>             //Timer library functions
#include <delays.h>             //Delay library functions
#include <i2c.h>                //I2C library functions
#include <stdlib.h>             //Library functions
#include <usart.h>              //USART library functions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Configurações para gravação                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#pragma config FOSC = HS
#pragma config CPUDIV = OSC1_PLL2
#pragma config WDT = OFF
#pragma config WDTPS = 128
#pragma config LVP = OFF
#pragma config PWRT = ON
#pragma config BOR = OFF
#pragma config BORV = 0
#pragma config PBADEN = OFF
```

```
#pragma config DEBUG = OFF
#pragma config PLLDIV = 1
#pragma config USBDIV = 1
#pragma config FCMEN = OFF
#pragma config IESO = OFF
#pragma config VREGEN = OFF
#pragma config MCLRE = ON
#pragma config LPT1OSC = OFF
#pragma config CCP2MX = ON
#pragma config STVREN = OFF
#pragma config ICPRT = OFF

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Definição e inicialização das variáveis Globais           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.
unsigned char    CONTADOR;        //BASE DE TEMPO PARA A PISCADA
unsigned char    TEMPO1;          //REGISTRADORES AUXILIARES DE TEMPO

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Constantes internas                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de constantes facilita a programação e a manutenção.
#define MIN         10
#define MAX         240
#define STEP        5
#define MULTIPLO    5

//A CONSTANTE DISPLAY REPRESENTA O SÍMBOLO QUE APARECERÁ PISCANDO NO
//DISPLAY. 1=LED LIGADO E 0=LED DESLIGADO. A RELAÇÃO ENTRE BITS E
//SEGMENTOS É A SEGUINTE: '.GFEDCBA'
//          a
//     **********
//     *        *
//   f *        * b
//     *    g   *
//     **********
//     *        *
//   e *        * c
//     *    d   *
//     **********  *.

#define     DISPLAY     0b01110110     //(LETRA H)

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Declaração dos flags de software                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de flags ajuda na programação e economiza memória RAM.
//Este programa não utiliza nenhum flag de usuário

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Definição e inicialização dos port's                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       PROTOTIPAGEM DE FUNÇÕES                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              ENTRADAS                                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//As entradas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define     BT1     PORTBbits.RB0   //PORTA DO BOTÃO
                                    //1 -> PRESSIONADO
```

```
                                    //0 -> LIBERADO

#define     BT2     PORTBbits.RB1   //PORTA DO BOTÃO
                                    //1 -> PRESSIONADO
                                    //0 -> LIBERADO

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                                SAÍDAS                                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//As saídas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define     DSP_UNIDADE     PORTBbits.RB7   //PORTA DO LED
                                            //0 -> APAGADO
                                            //1 -> ACESO

#define     LINHA_4         PORTBbits.RB7   //PINO PARA ATIVAR LINHA 4 DO TECLADO
                                            //MATRICIAL
                                            //1 -> LINHA ATIVADA
                                            //0 -> LINHA DESATIVADA

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função Principal                             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void main ()
{
    PORTA = 0x00;               //Limpa PORTA
    PORTB = 0x00;               //Limpa PORTB
    PORTC = 0x00;               //Limpa PORTC
    PORTD = 0x00;               //Limpa PORTD
    PORTE = 0x00;               //Limpa PORTE

    LATA = 0x00;                //Limpa PORTA
    LATB = 0x00;                //Limpa PORTB
    LATC = 0x00;                //Limpa PORTC
    LATD = 0x00;                //Limpa PORTD
    LATE = 0x00;                //Limpa PORTE

    TRISA = 0b00101111;         //CONFIG DIREÇÃO DOS PINOS PORTA
    TRISB = 0b00001111;         //CONFIG DIREÇÃO DOS PINOS PORTB
    TRISC = 0b10011111;         //CONFIG DIREÇÃO DOS PINOS PORTC
    TRISD = 0b00000000;         //CONFIG DIREÇÃO DOS PINOS PORTD
    TRISE = 0b00000111;         //CONFIG DIREÇÃO DOS PINOS PORTE

    ADCON1 = 0b00001111;        //DESLIGA CONVERSORES A/D

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Inicialização do Sistema                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
    DSP_UNIDADE = 1;
    PORTD = DISPLAY;            //ACENDE O VALOR CERTO NO DISPLAY
    CONTADOR = MIN;             //INICIA CONTADOR COM VALOR MIN.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Rotina Principal                             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
    while(1)
    {
        ClrWdt();
        TEMPO1 = MULTIPLO;
        do
        {
            Delay1KTCYx(CONTADOR);
            if (BT1)
            {
```

```
                CONTADOR = CONTADOR - STEP; //INCREMENTA O CONTADOR EM STEP
                if (CONTADOR < MIN)
                {
                    CONTADOR = MIN;
                    while(BT1) ClrWdt();
                }
                continue;
            }

            if (BT2)
            {
                CONTADOR = CONTADOR + STEP; //INCREMENTA O CONTADOR EM STEP
                if (CONTADOR >= MAX)
                {
                    CONTADOR = MAX;
                    while(BT2) ClrWdt();
                }
                continue;
            }
        }while(--TEMPO1);                   //VOLTA PARA O INICIO ENQUANTO
                                            //TEMPO1 DIFERENTE DE ZERO
        PORTD = PORTD ^ DISPLAY;            //INVERTE O PORTD
    }

}   // FIM DO PROGRAMA
```

5.6. Dicas e Comentários

A rotina de delay recebe o argumento passado pelo WORK para determinar o delay que deve ser gerado, no entanto, o valor máximo assumido pelo WORK é 255 e conforme comentado o delay máximo gerado é de 256ms. Como isto é possível?

Acontece que a rotina foi escrita de forma que ela é executada pelo menos uma vez. Assim, a rotina executa um delay de 1ms, decrementa o argumento e quando este é igual a zero retorna. Porém, quando o argumento passado pelo WORK é zero, ao primeiro decremento, o argumento estoura e pula para 255. Como a condição é testada após o decremento e como após o decremento o valor do argumento é 255, o delay continua a ser gerado, por mais 255 vezes e desta forma obtêm-se um delay de 256ms.

A proposta da rotina de delay foi gerar um delay fundamental em 1ms, porém, a mesma idéia pode ser utilizada para criar rotinas com delays fundamentais diferentes.

5.7. Exercícios Propostos

1. Alterar a rotina de delay para gerar um delay fundamental de 100us.
2. Alterar a rotina de delay para gerar um delay fundamental de 10ms.
3. Incluir o tratamento de debounce nas teclas

6. Experiência 4 –Conversão BCD para displays de 7 segmentos

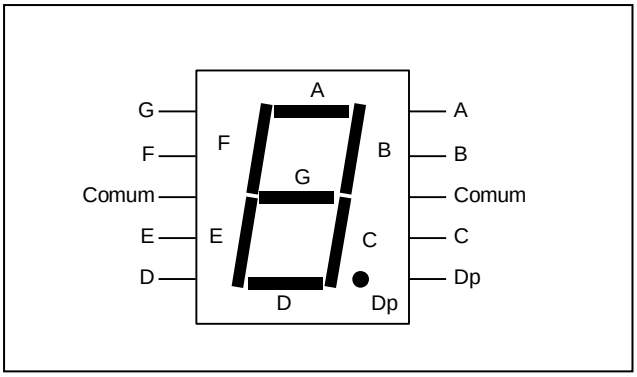
6.1. Objetivo

O objetivo desta experiência é ensinar ao aluno como criar um decodificar BCD para displays de 7 segmentos.

6.2. Descrição

Os displays utilizados no McMaster são conhecidos como displays de leds de 7 segmentos, pois os números são compostos por 7 traços. Estes componentes possuem ainda o ponto decimal e são considerados displays numéricos, por não possuírem traços suficientes para a exibição de todas as letras do nosso alfabeto.

Para facilitar a vida do projetista o mercado padronizou uma nomenclatura para todos os traços do display, possibilitando que tratemos cada um deles individualmente:



Desta forma, temos um pino para controlar cada um dos segmentos (A...G) e mais o ponto decimal (Dp). Os dois pinos adicionais são os comuns, que podem ser ligados a todos os catodos ou anodos dos leds internos. Por causa disso, estes displays são fornecidos em 2 tipos: catodo comum ou anodo comum. No nosso caso, os displays utilizados são do tipo catodo comum, isto é, o pino comum deve ser ligado ao terra e os segmentos devem ser ligados ao Vcc para acenderem.

Outra observação importante é que a pinagem descrita no desenho é válida para o tipo de display utilizado no McMaster. Existem displays de outros tamanhos que possuem uma disposição de pinos diferente.

Como cada segmento é um led individual, precisa-se de um pino do PIC para controlar cada segmento. Desta forma, são necessários 8 pinos para acionar os 7 segmentos e mais o ponto decimal.

A fim de converter o valor binário de um algarismo em um valor de 8 bits que represente o este algarismo num display de 7 segmentos fez-se uso de uma tabela de conversão. Por exemplo, para representar o algarismo “2” no display, deve-se acender os segmentos A, B, D, E e G.

Como no kit McMaster o PORTD está conectado aos segmentos conforme a tabela

PIC	Segmento
RD0	A
RD1	B
RD2	C
RD3	D
RD4	E
RD5	F
RD6	G
RD7	DP

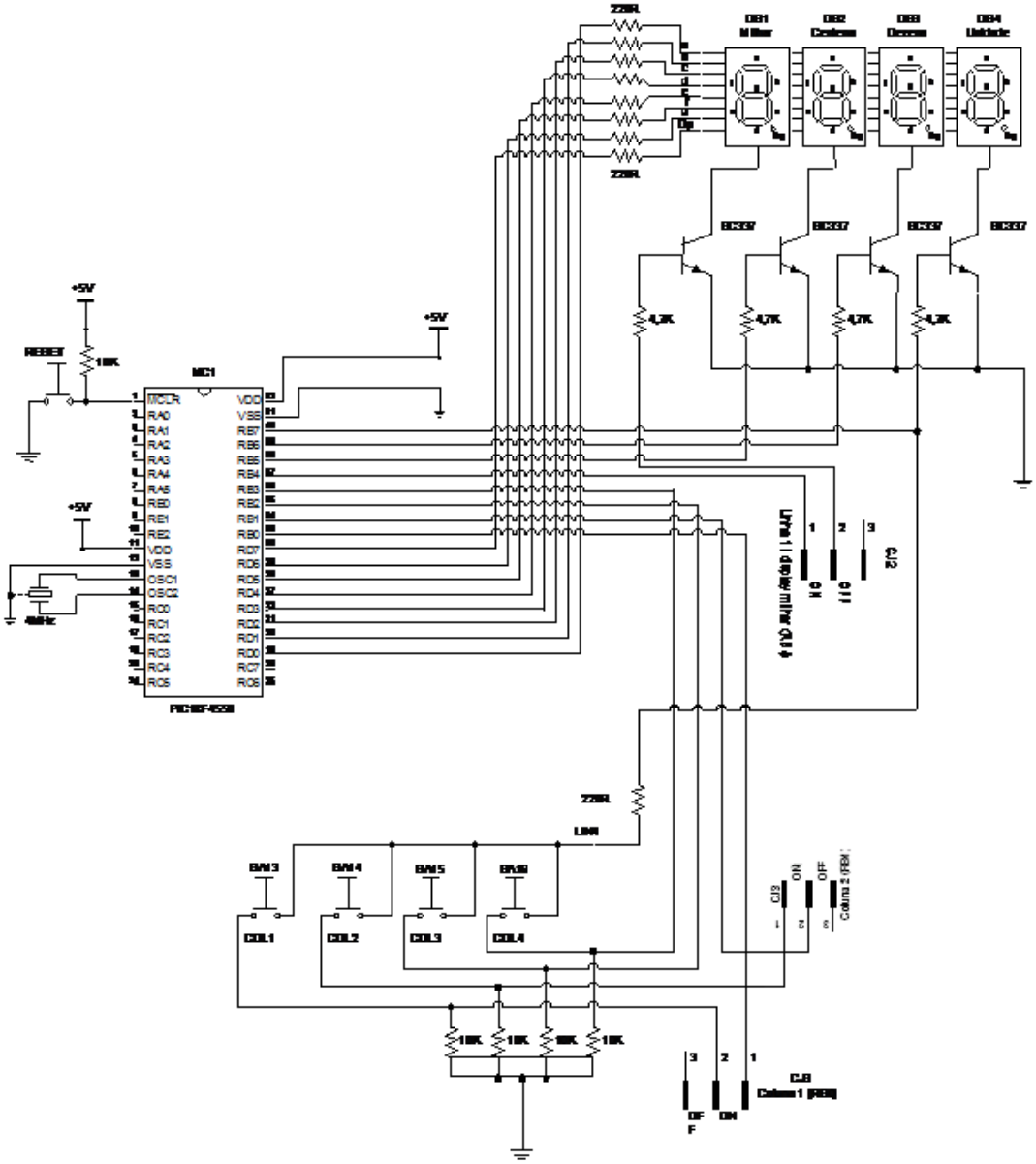
para acender o algarismo “2” no display precisa-se colocar o valor 01011011b no PORTD, ou seja, colocar em 1 os bits ligados aos segmentos que se deseja acender.

Assim, no software deve ser criada uma tabela para converter cada valor binário numa representação que posta nos segmentos (PORTD) reproduza o valor binário original.

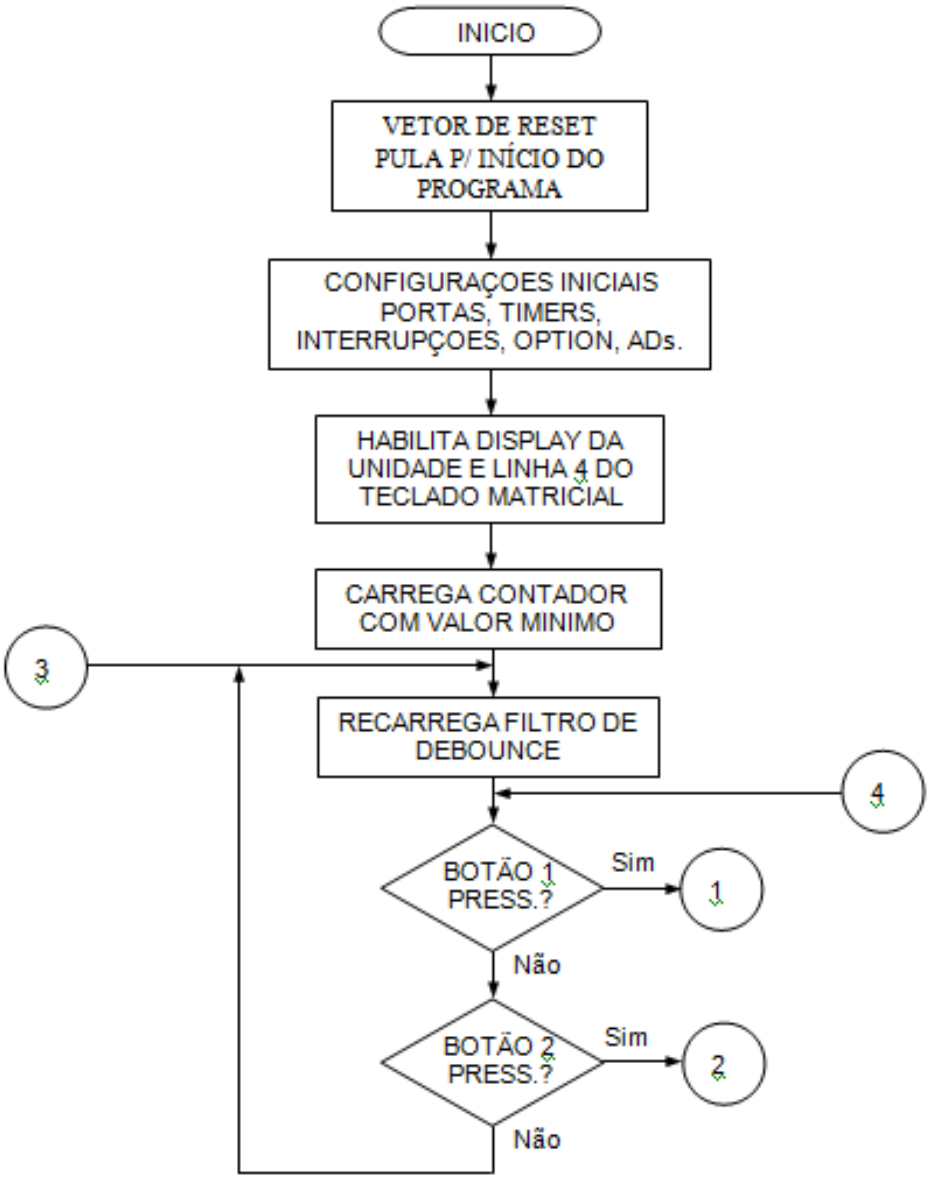
A tabela é criada utilizando-se a instrução RETLW. O valor binário que se deseja converter é adicionado ao PCL (program counter) de forma a desviar o fluxo do programa para a linha que contém a combinação de 0 e 1 que formarão o caractere no display.

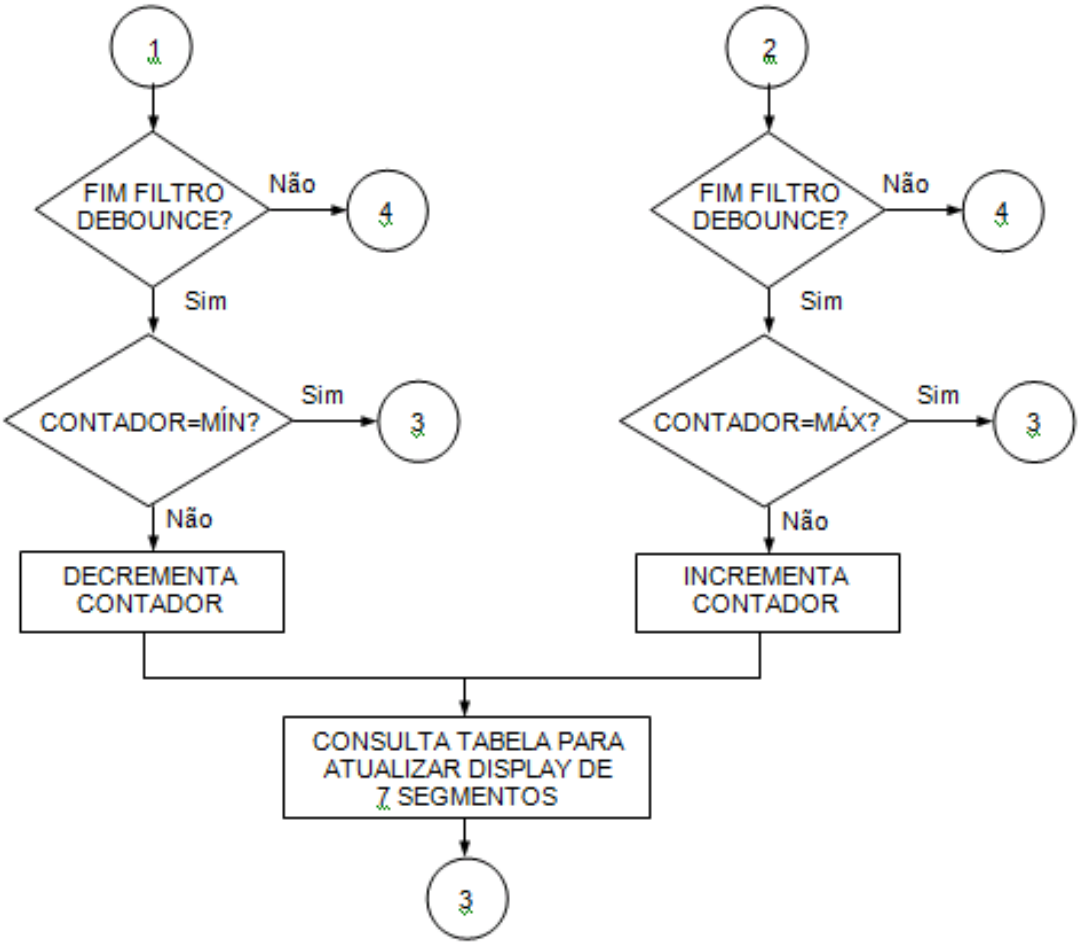
O software utiliza as teclas da linha 4 colunas 1 e 2 para incrementar e decrementar o valor da variável “CONTADOR”. Esta variável está limitada pelas constantes "MIN" e "MAX". A tabela de conversão foi utilizada a fim de visualizar o valor da variável “CONTADOR” no display de 7 segmentos.

6.3. *Esquema Elétrico*



6.4. Fluxograma




1
FIM FILTRO DEBOUNCE?
Não
4
Sim
CONTADOR=MÍN?
Sim
3
Não
DECREMENTA CONTADOR
2
FIM FILTRO DEBOUNCE?
Não
4
Sim
CONTADOR=MÁX?
Sim
3
Não
INCREMENTA CONTADOR
CONSULTA TABELA PARA ATUALIZAR DISPLAY DE 7 SEGMENTOS
3

6.5. Código

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Exemplo de Código para utilização com a McMaster             *
 *                                 Exemplo 4                                 *
 *                                                                           *
 *                                  MOSAICO                                  *
 *                                                                           *
 *  TEL: (0XX11) 4992-8775                 SITE: www.mosaico.com.br          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                            *
 *   DATA : 18/01/2010                                                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              Descrição geral                              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

//SOFTWARE QUE UTILIZA DOIS BOTÕES PARA INCREMENTAR E DECREMENTAR UM
//CONTADOR. O VALOR É MOSTRADO NO DISPLAY DE 7 SEGMENTOS.

/*
 CONFIGURAÇÂO DOS JUMPERS DA PLACA:

        CAP USB(RC3)                -> ON
        DATA I2C                    -> OFF
        CLOCK I2C                   -> OFF
        COLUNA 2(RB1)               -> ON
        COLUNA 1(RB0)               -> ON
        LINHA 1 / DISPLAY MILHAR(RB4)    -> ON
        LEDS ESPECIAS (RC0,RC1,RC2)      -> ON

*/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                             DEFINIÇÃO PIC                                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <p18F4550.h>           // Register definitions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               INCLUDES DAS FUNÇÕES DE PERIFÉRICOS DO PIC                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <pwm.h>                //PWM library functions
#include <adc.h>                //ADC library functions
#include <timers.h>             //Timer library functions
#include <delays.h>             //Delay library functions
#include <i2c.h>                //I2C library functions
#include <stdlib.h>             //Library functions
#include <usart.h>              //USART library functions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Configurações para gravação                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#pragma config FOSC = HS
#pragma config CPUDIV = OSC1_PLL2
#pragma config WDT = OFF
#pragma config WDTPS = 128
#pragma config LVP = OFF
```

```
#pragma config PWRT = ON
#pragma config BOR = OFF
#pragma config BORV = 0
#pragma config PBADEN = OFF
#pragma config DEBUG = OFF
#pragma config PLLDIV = 1
#pragma config USBDIV = 1
#pragma config FCMEN = OFF
#pragma config IESO = OFF
#pragma config VREGEN = OFF
#pragma config MCLRE = ON
#pragma config LPT1OSC = OFF
#pragma config CCP2MX = ON
#pragma config STVREN = OFF
#pragma config ICPRT = OFF

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Definição e inicialização das variáveis Globais          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.
unsigned char   CONTADOR;       //ARMAZENA O VALOR DA CONTAGEM
unsigned char   FILTRO11;       //FILTRAGEM  1 PARA O BOTÃO 1
unsigned char   FILTRO12;       //FILTRAGEM  2 PARA O BOTÃO 1
unsigned char   FILTRO21;       //FILTRAGEM  1 PARA O BOTÃO 2
unsigned char   FILTRO22;       //FILTRAGEM  2 PARA O BOTÃO 2

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Constantes internas                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de constantes facilita a programação e a manutenção.
#define     MIN         0
#define     MAX         15
#define     T_FILTRO    5

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Declaração dos flags de software                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de flags ajuda na programação e economiza memória RAM.
struct
{
    unsigned BIT0:1;
    unsigned BIT1:1;
    unsigned BIT2:1;
    unsigned BIT3:1;
    unsigned BIT4:1;
    unsigned BIT5:1;
    unsigned BIT6:1;
    unsigned BIT7:1;
}FLAGSbits;             //ARMAZENA OS FLAGS DE CONTROLE

#define     ST_BT1    FLAGSbits.BIT0        //STATUS DO BOTÃO 1
#define     ST_BT2    FLAGSbits.BIT1        //STATUS DO BOTÃO 2

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Definição e inicialização dos port's               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       PROTOTIPAGEM DE FUNÇÕES                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                               ENTRADAS                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//As entradas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.
```

```
#define     BT1     PORTBbits.RB0   //PORTA DO BOTÃO
                                    //1 -> PRESSIONADO
                                    //0 -> LIBERADO

#define     BT2     PORTBbits.RB1   //PORTA DO BOTÃO
                                    //1 -> PRESSIONADO
                                    //0 -> LIBERADO

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              SAÍDAS                                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//As saídas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define     LINHA_4     PORTBbits.RB7   //PINO PARA ATIVAR LINHA 4 DO TECLADO
                                        //MATRICIAL
                                        //1 -> LINHA ATIVADA
                                        //0 -> LINHA DESATIVADA

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
// *                  ROTINA DE CONVERSÃO BINÁRIO -> DISPLAY              *
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
// ESTA ROTINA IRÁ RETORNAR EM W, O SIMBOLO CORRETO QUE DEVE SER
// MOSTRADO NO DISPLAY PARA CADA VALOR DE INTENSIDADE. O RETORNO JÁ ESTÁ
// FORMATADO PARA AS CONDIÇÕES DE LIGAÇÃO DO DISPLAY AO PORTD.
//          a
//     **********
//     *        *
//   f *        * b
//     *    g   *
//     **********
//     *        *
//   e *        * c
//     *    d   *
//     **********  *.

const rom unsigned char CONVERTE[16] =
{//   .GFEDCBA       POSIÇÃO CORRETA DOS SEGMENTOS
    0b00111111,     // 00 - RETORNA SÍMBOLO CORRETO 0
    0b00000110,     // 01 - RETORNA SÍMBOLO CORRETO 1
    0b01011011,     // 02 - RETORNA SÍMBOLO CORRETO 2
    0b01001111,     // 03 - RETORNA SÍMBOLO CORRETO 3
    0b01100110,     // 04 - RETORNA SÍMBOLO CORRETO 4
    0b01101101,     // 05 - RETORNA SÍMBOLO CORRETO 5
    0b01111101,     // 06 - RETORNA SÍMBOLO CORRETO 6
    0b00000111,     // 07 - RETORNA SÍMBOLO CORRETO 7
    0b01111111,     // 08 - RETORNA SÍMBOLO CORRETO 8
    0b01101111,     // 09 - RETORNA SÍMBOLO CORRETO 9
    0b01110111,     // 10 - RETORNA SÍMBOLO CORRETO A
    0b01111100,     // 11 - RETORNA SÍMBOLO CORRETO b
    0b00111001,     // 12 - RETORNA SÍMBOLO CORRETO C
    0b01011110,     // 13 - RETORNA SÍMBOLO CORRETO d
    0b01111001,     // 14 - RETORNA SÍMBOLO CORRETO E
    0b01110001};    // 15 - RETORNA SÍMBOLO CORRETO F

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função Principal                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void main ()
{
    PORTA = 0x00;               //Limpa PORTA
    PORTB = 0x00;               //Limpa PORTB
    PORTC = 0x00;               //Limpa PORTC
    PORTD = 0x00;               //Limpa PORTD
```

```
    PORTE = 0x00;               //Limpa PORTE

    LATA = 0x00;                //Limpa PORTA
    LATB = 0x00;                //Limpa PORTB
    LATC = 0x00;                //Limpa PORTC
    LATD = 0x00;                //Limpa PORTD
    LATE = 0x00;                //Limpa PORTE

    TRISA = 0b00101111;         //CONFIG DIREÇÃO DOS PINOS PORTA
    TRISB = 0b00001111;         //CONFIG DIREÇÃO DOS PINOS PORTB
    TRISC = 0b10011001;         //CONFIG DIREÇÃO DOS PINOS PORTC
    TRISD = 0b00000000;         //CONFIG DIREÇÃO DOS PINOS PORTD
    TRISE = 0b00000000;         //CONFIG DIREÇÃO DOS PINOS PORTE

    ADCON1 = 0b00001111;        //DESLIGA CONVERSORES A/D

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      Inicialização do Sistema                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

    LINHA_4 = 1;        //ATIVA A LINHA 4 DO TECLADO E DISPLAY DA UNIDADE

    ST_BT1 = 0;
    ST_BT2 = 0;
    CONTADOR = MIN;                 //INICIA CONTADOR = MIN
    PORTD = CONVERTE[CONTADOR];     //ATUALIZA O DISPLAY INICIALMENTE

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Rotina Principal                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
    while(1)
    {
        ClrWdt();
        FILTRO11 = 0;
        FILTRO21 = 0;
        FILTRO12 = T_FILTRO;
        FILTRO22 = T_FILTRO;

        while (BT1)
        {
            ClrWdt();
            if (--FILTRO11 == 0)
            {
                FILTRO12--;
                if(FILTRO12 == 0)
                {
                    if (CONTADOR != MIN)
                    {
                        if (!ST_BT1)
                        {
                            CONTADOR--;
                            PORTD = CONVERTE[CONTADOR];
                            ST_BT1 = 1;
                        }
                    }
                }
            }
        }
        ST_BT1 = 0; // BOTÃO LIBERADO, LIMPA O FLAG

        while (BT2)
        {
            ClrWdt();
            if (--FILTRO21 == 0)
            {
                FILTRO22--;
```

```
                if(FILTRO22 == 0)
                {
                    if (CONTADOR != MAX)
                    {
                        if (!ST_BT2)
                        {
                            CONTADOR++;
                            PORTD = CONVERTE[CONTADOR];
                            ST_BT2 = 1;
                        }
                    }
                }
            }
        }
        ST_BT2 = 0; // BOTÃO LIBERADO, LIMPA O FLAG
    }
}   // FIM DO PROGRAMA
```

6.6. Dicas e Comentários

A tabela implementada neste exemplo apresenta 16 linhas, assim, é possível representar qualquer valor hexadecimal entre 0h e Fh. No entanto, utilizando o mesmo conceito poderíamos criar representações para praticamente todas as letras do alfabeto. E para escrever mensagens no display bastaria associar a cada letra um número binário.

O conceito da tabela não se limita apenas à utilização com displays de 7 segmentos, infinitas situações podem ser resolvidas com o uso de uma tabela. Por exemplo, pode-se criar uma tabela para converter números binários em caracteres ASCII e vice-versa.

6.7. Exercícios Propostos

1. Mantendo as 16 linhas atuais, alterar a tabela para mostrar as letras: “A”, “b”, “C”, “d”, “E”, “F”, “G”, “H”, “I”, “J”, “L”, “n”, “O”, “P”, “r” e “S”.
2. Alterar o software a fim de utilizar uma tabela que contenha todos os números e todas as possíveis letras.

7. Experiência 5 – Timer de segundos

7.1. Objetivo

O uso de rotinas de delays para contagem de tempo nem sempre pode ser aplicado, uma vez que este tipo de rotina deixa o processador parado. Um recurso muito mais adequado para a contagem de tempos é a utilização do timer do microcontrolador. Este é o objetivo desta experiência. Ensinar ao aluno como configurar e utilizar o timer e a interrupção.

7.2. Descrição

Esta experiência cria um timer decrescente em segundos. O valor inicial é determinado pela constante V_INICIO e pode estar entre 1 e 15 segundos.

Os botões ativos são os da linha 4. O botão da coluna 1 dispara o timer, mostrando o tempo restante no display. O da coluna 2 paralisa o timer. O led ligado ao pino RC0 é utilizado para indicar o estado atual do timer sendo acesso se o timer estiver rodando e apagado se o timer estiver parado.

O timer utiliza como base de tempo a interrupção de TMR0 sendo que esta ocorre quando o timer estoura, ou seja, quando o valor do TMR0 pula de 0xFF para 0x00. Assim, o intervalo de tempo entre interrupções irá depender do prescaler configurado para o timer, do valor inicial com que ele é carregado e do tempo de execução de um ciclo de máquina.

No exemplo desta experiência, sempre que se entra na interrupção de TMR0, o contador do timer é carregado com 131, de forma que sempre se faça 125 contagens, pois, o timer irá contar de 131 até 256 e quando voltar a estourar será novamente carregado com 131.

Desta forma, podemos calcular o tempo entre interrupções seguindo a equação abaixo:

Tempo TMR0 = (256 – valor com que é carregado) * prescaler * ciclo de máquina

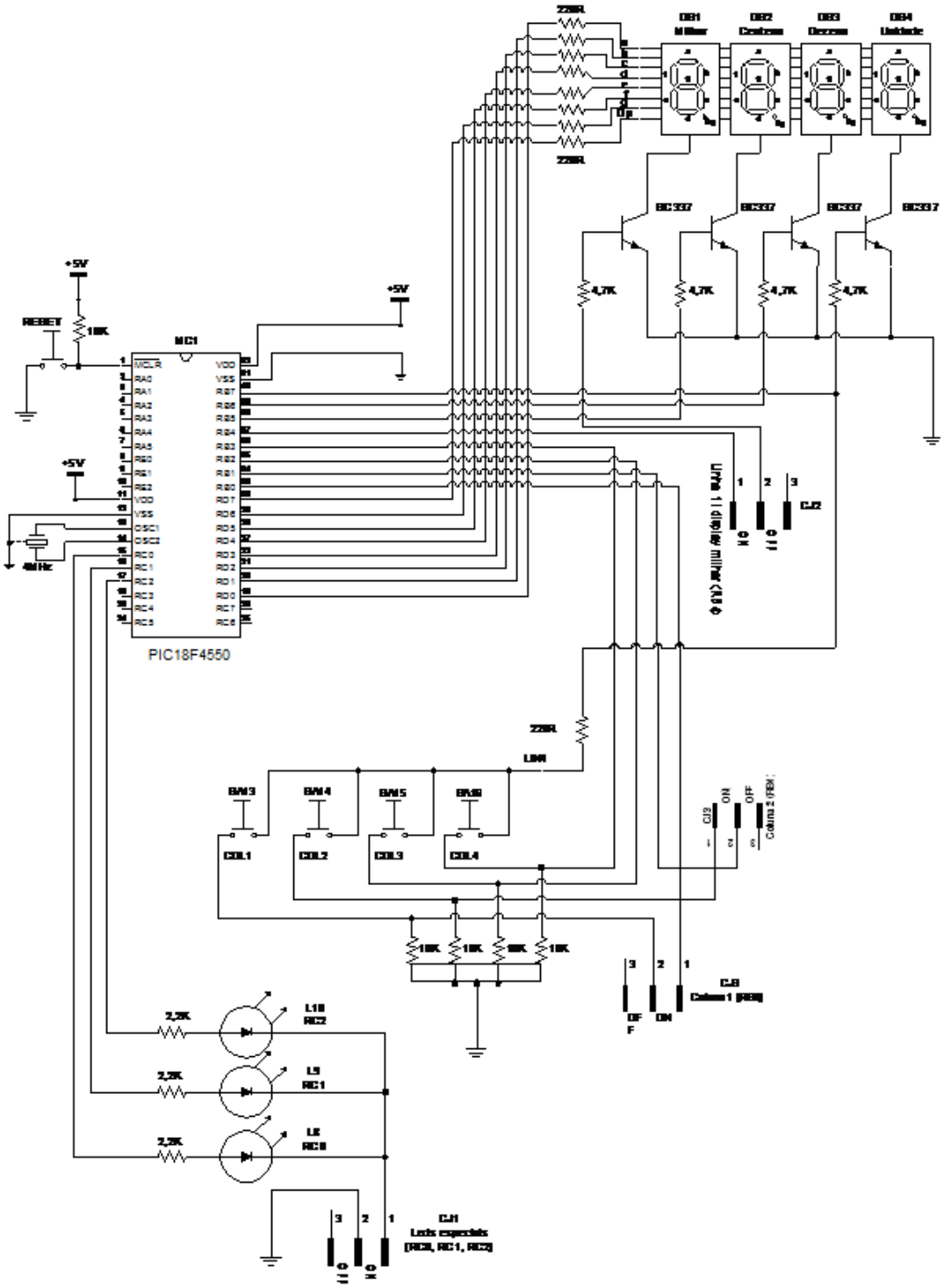
Tempo TMR0 = (256 – 131) * 64 * 1us

Tempo TMR0 = 8ms

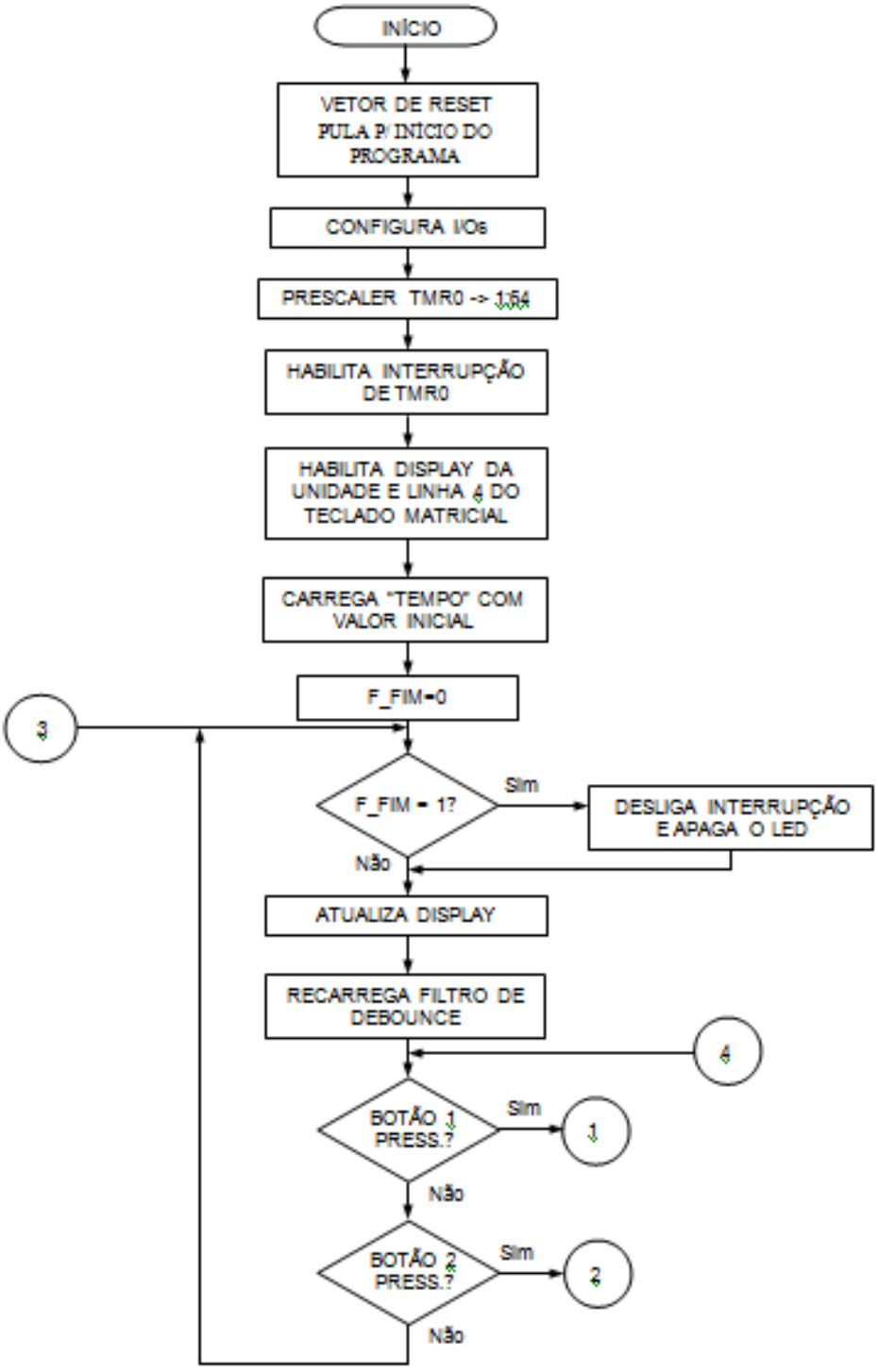
Portanto, a interrupção irá ocorrer a cada 8ms.

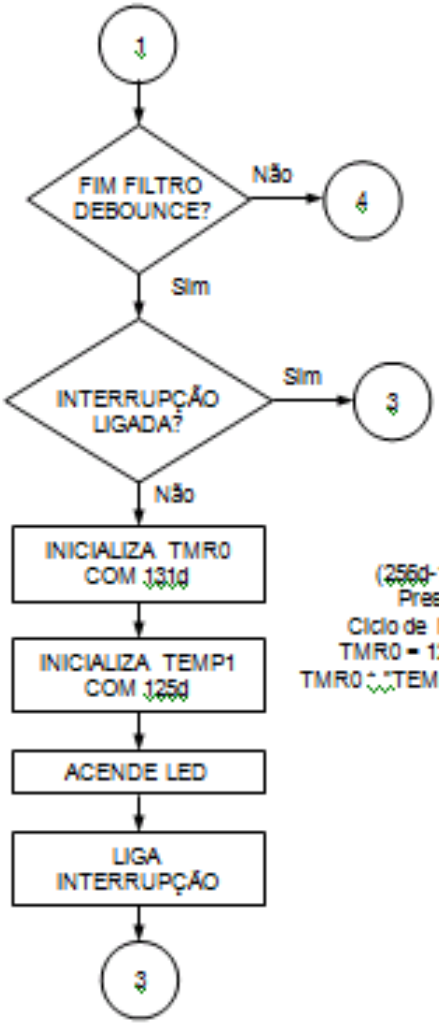
A fim de criar a base de tempo de 1 segundo, foi acrescentado um contador auxiliar que conta o número de ocorrências da interrupção de TMR0. Com este contador auxiliar pode-se estender a base de tempo sempre em múltiplos de 8ms. Caso este contador auxiliar seja configurado para contar 125 interrupções de TMR0, pode-se obter a base de tempo de 1 segundo, fundamental para a criação do timer da experiência.

7.3. Esquema Elétrico

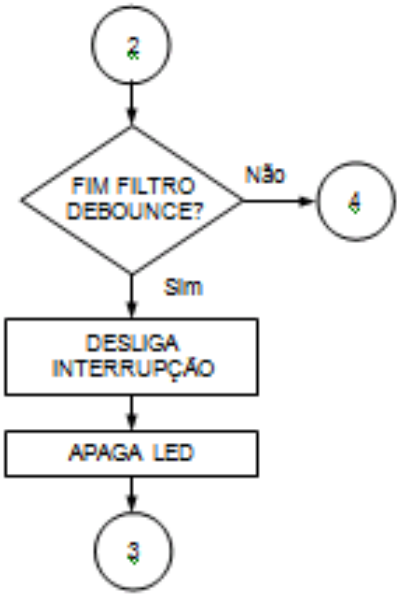


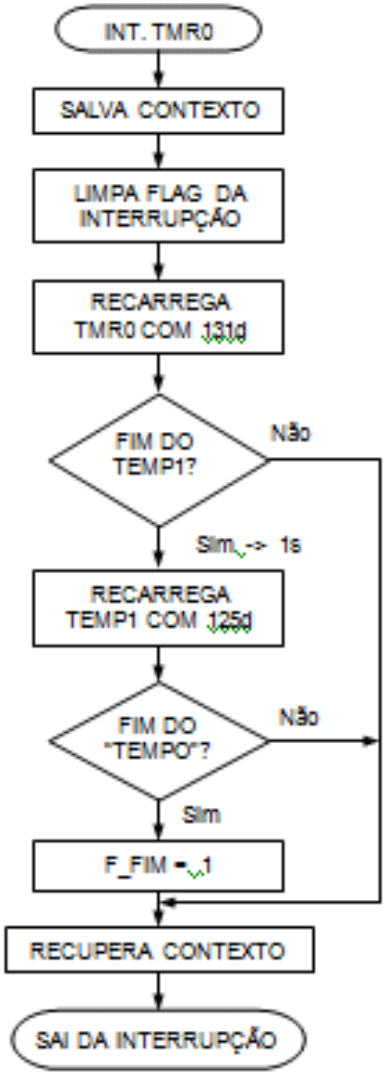
7.4. Fluxograma





(256d-125d)=131d
Prescale 1:64
Ciclo de Máquina = 1us
TMR0 = 125*64us = 8ms
TMR0 * "TEMP1" = 8ms * 125 =




INT. TMR0
SALVA CONTEXTO
LIMPA FLAG DA INTERRUPÇÃO
RECARREGA TMR0 COM 131d
FIM DO TEMP1?
Não
Sim -> 1s
RECARREGA TEMP1 COM 125d
FIM DO "TEMPO"?
Não
Sim
F_FIM = 1
RECUPERA CONTEXTO
SAI DA INTERRUPÇÃO

7.5. Código

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Exemplo de Código para utilização com a McMaster            *
 *                              Exemplo 5                                  *
 *                                                                         *
 *                               MOSAICO                                   *
 *                                                                         *
 *  TEL: (0XX11) 4992-8775                SITE: www.mosaico.com.br         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                          *
 *   DATA : 18/01/2010                                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Descrição geral                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*
TIMER DECRESCENTE EM SEGUNDOS. O VALOR INICIAL É DETERMINADO PELA CONSTANTE
V_INICIO E PODE ESTAR ENTRE 1 E 15 SEGUNDOS. OS BOTÕES ATIVOS SÃO O DA
LINHA 4. O BOTÃO DA COLUNA 1 DISPARA O TIMER, MOSTRANDO O TEMPO RESTANTE
NO DISPLAY. O DA COLUNA 2 PARALIZA O TIMER. O LED (RC0) É UTILIZADO PARA
INDICAR O ESTADO ATUAL DO TIMER: ACESO=RODANDO E APAGADO=PARADO
*/
/*
 CONFIGURAÇÂO DOS JUMPERS DA PLACA:

        CAP USB(RC3)              -> ON
        DATA I2C                  -> OFF
        CLOCK I2C                 -> OFF
        COLUNA 2(RB1)             -> ON
        COLUNA 1(RB0)             -> ON
        LINHA 1 / DISPLAY MILHAR(RB4)    -> ON
        LEDS ESPECIAS (RC0,RC1,RC2)      -> ON

*/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            DEFINIÇÃO PIC                                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <p18F4550.h>           // Register definitions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               INCLUDES DAS FUNÇÕES DE PERIFÉRICOS DO PIC                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <pwm.h>                //PWM library functions
#include <adc.h>                //ADC library functions
#include <timers.h>             //Timer library functions
#include <delays.h>             //Delay library functions
#include <i2c.h>                //I2C library functions
#include <stdlib.h>             //Library functions
#include <usart.h>              //USART library functions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Configurações para gravação                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#pragma config FOSC = HS
#pragma config CPUDIV = OSC1_PLL2
#pragma config WDT = OFF
#pragma config WDTPS = 128
#pragma config LVP = OFF
#pragma config PWRT = ON
#pragma config BOR = OFF
```

```
#pragma config BORV = 0
#pragma config PBADEN       = OFF
#pragma config DEBUG        = OFF
#pragma config PLLDIV       = 1
#pragma config USBDIV       = 1
#pragma config FCMEN        = OFF
#pragma config IESO  = OFF
#pragma config VREGEN       = OFF
#pragma config MCLRE        = ON
#pragma config LPT1OSC      = OFF
#pragma config CCP2MX = ON
#pragma config STVREN       = OFF
#pragma config ICPRT        = OFF


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Definição e inicialização das variáveis Globais           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.
unsigned char   TEMPO;          //ARMAZENA O VALOR DO TEMPO
unsigned char   FILTRO11;       //FILTRAGEM  1 PARA O BOTÃO 1
unsigned char   FILTRO12;       //FILTRAGEM  2 PARA O BOTÃO 1
unsigned char   FILTRO21;       //FILTRAGEM  1 PARA O BOTÃO 2
unsigned char   FILTRO22;       //FILTRAGEM  2 PARA O BOTÃO 2
unsigned char   TEMP1;          //REGISTRADORES AUXILIARES
unsigned char   TEMP2;          //

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                         Constantes internas                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de constantes facilita a programação e a manutenção.
#define     V_INICIO    15
#define     T_FILTRO    255

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Declaração dos flags de software                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de flags ajuda na programação e economiza memória RAM.
struct
{
    unsigned BIT0:1;
    unsigned BIT1:1;
    unsigned BIT2:1;
    unsigned BIT3:1;
    unsigned BIT4:1;
    unsigned BIT5:1;
    unsigned BIT6:1;
    unsigned BIT7:1;
}FLAGSbits;             //ARMAZENA OS FLAGS DE CONTROLE

#define     ST_BT1      FLAGSbits.BIT0          //STATUS DO BOTÃO 1
#define     ST_BT2      FLAGSbits.BIT1          //STATUS DO BOTÃO 2
#define     F_FIM       FLAGSbits.BIT2          //FLAG DE FIM DE TEMPO

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Definição e inicialização dos port's                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      PROTOTIPAGEM DE FUNÇÕES                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void TRATA_INT_TIMER0 (void);

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                               ENTRADAS                                *
```

```
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//As entradas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define     BT1     PORTBbits.RB0   //PORTA DO BOTÃO
                                    //1 -> PRESSIONADO
                                    //0 -> LIBERADO

#define     BT2     PORTBbits.RB1   //PORTA DO BOTÃO
                                    //1 -> PRESSIONADO
                                    //0 -> LIBERADO

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                                SAÍDAS                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//As saídas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define     LINHA_4     LATBbits.LATB7   //PINO PARA ATIVAR LINHA 4 DO TECLADO
                                        //MATRICIAL
                                        //1 -> LINHA ATIVADA
                                        //0 -> LINHA DESATIVADA

#define     LED         LATCbits.LATC0  //DEFINE PINO DO LED
                                        //0 -> LED APAGADO
                                        //1 -> LED ACESO

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
// *                 ROTINA DE TRATAMENTO DE INT DE TIMER0                 *
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
#pragma code TIMER0_interrupt = 0x0008
void TIMER0_int (void)
{
  _asm goto TRATA_INT_TIMER0 _endasm
}
#pragma code

#pragma interrupt TRATA_INT_TIMER0
void TRATA_INT_TIMER0 (void)
{
    INTCONbits.TMR0IF = 0;                          //LIMPA FLAG DE INTERRUPÇÃO
    WriteTimer0(256 - 125);
    TEMP1--;
    if (TEMP1 == 0)
    {
         TEMP1 = 125;
         if(F_FIM == 1)
              return;
         TEMPO--;
         if(TEMPO == 0) F_FIM = 1;
    }

}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
// *                   ROTINA DE CONVERSÃO BINÁRIO -> DISPLAY              *
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
// ESTA ROTINA IRÁ RETORNAR EM W, O SIMBOLO CORRETO QUE DEVE SER
// MOSTRADO NO DISPLAY PARA CADA VALOR DE INTENSIDADE. O RETORNO JÁ ESTÁ
// FORMATADO PARA AS CONDIÇÕES DE LIGAÇÃO DO DISPLAY AO PORTD.
//          a
//     **********
//     *        *
//   f *        * b
//     *    g   *
//     **********
```

```
//     *         *
//   e *         * c
//     *    d    *
//     **********  *.

const rom unsigned char CONVERTE[16] =
{//   .GFEDCBA       POSIÇÃO CORRETA DOS SEGMENTOS
    0b00111111,     // 00 - RETORNA SÍMBOLO CORRETO 0
    0b00000110,     // 01 - RETORNA SÍMBOLO CORRETO 1
    0b01011011,     // 02 - RETORNA SÍMBOLO CORRETO 2
    0b01001111,     // 03 - RETORNA SÍMBOLO CORRETO 3
    0b01100110,     // 04 - RETORNA SÍMBOLO CORRETO 4
    0b01101101,     // 05 - RETORNA SÍMBOLO CORRETO 5
    0b01111101,     // 06 - RETORNA SÍMBOLO CORRETO 6
    0b00000111,     // 07 - RETORNA SÍMBOLO CORRETO 7
    0b01111111,     // 08 - RETORNA SÍMBOLO CORRETO 8
    0b01101111,     // 09 - RETORNA SÍMBOLO CORRETO 9
    0b01110111,     // 10 - RETORNA SÍMBOLO CORRETO A
    0b01111100,     // 11 - RETORNA SÍMBOLO CORRETO b
    0b00111001,     // 12 - RETORNA SÍMBOLO CORRETO C
    0b01011110,     // 13 - RETORNA SÍMBOLO CORRETO d
    0b01111001,     // 14 - RETORNA SÍMBOLO CORRETO E
    0b01110001};    // 15 - RETORNA SÍMBOLO CORRETO F


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      ROTINA DE ATUALIZAÇÃO DO DISPLAY                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// ESTA ROTINA CONVERTE O VALOR DE TEMPO ATRAVÉS DA ROTINA CONVERTE
// E ATUALIZA O PORTB PARA ACENDER O DISPLAY CORRETAMENTE

void ATUALIZA(void)
{
    LATD = CONVERTE[TEMPO];         //CONVERTE CONTADOR NO NÚMERO DO
                            //DISPLAY E ATUALIZA O PORTD PARA
                            //VISUALIZARMOS O VALOR DE CONTADOR
                            //NO DISPLAY
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      ROTINA DE DESLIGAR O TIMER                         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// ESTA ROTINA EXECUTA AS AÇÕES NECESSÁRIAS PARA DESLIGAR O TIMER

void DESL_TIMER(void)
{
    INTCONbits.GIE = 0;                 //DESLIGA CHAVE GERAL DE INT.
    LED = 0;                            //APAGA O LED
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      ROTINA DE LIGAR O TIMER                            *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// ESTA ROTINA EXECUTA AS AÇÕES NECESSÁRIAS PARA LIGAR O TIMER

void LIGA_TIMER(void)
{
    if(INTCONbits.GIE == 1)             //TIMER JÁ ESTA LIGADO?
    {

        return;                         //SIM, RETORNA DIRETO
    }

    INTCONbits.TMR0IF = 0;              //LIMPA FLAG DE INTERRUPÇÃO
    WriteTimer0(256 - 125);             //INICIA TIMER0 E TEMP1
    TEMP1 = 125;
```

```
    INTCONbits.GIE = 1;                 //LIGA CHAVE GERAL DE INT.
    LED = 1;                            //LIGA O LED
}
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função Principal                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void main ()
{
    PORTA = 0x00;               //Limpa PORTA
    PORTB = 0x00;               //Limpa PORTB
    PORTC = 0x00;               //Limpa PORTC
    PORTD = 0x00;               //Limpa PORTD
    PORTE = 0x00;               //Limpa PORTE

    LATA = 0x00;                //Limpa PORTA
    LATB = 0x00;                //Limpa PORTB
    LATC = 0x00;                //Limpa PORTC
    LATD = 0x00;                //Limpa PORTD
    LATE = 0x00;                //Limpa PORTE

    TRISA = 0b00101111;         //CONFIG DIREÇÃO DOS PINOS PORTA
    TRISB = 0b00001111;         //CONFIG DIREÇÃO DOS PINOS PORTB
    TRISC = 0b10011000;         //CONFIG DIREÇÃO DOS PINOS PORTC
    TRISD = 0b00000000;         //CONFIG DIREÇÃO DOS PINOS PORTD
    TRISE = 0b00000111;         //CONFIG DIREÇÃO DOS PINOS PORTE

    ADCON1 = 0b00001111;        //DESLIGA CONVERSORES A/D

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Inicialização do Sistema                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
    OpenTimer0(TIMER_INT_ON & T0_8BIT & T0_SOURCE_INT & T0_EDGE_FALL & T0_PS_1_64);

    LINHA_4 = 1;        //ATIVA A LINHA 4 DO TECLADO E DISPLAY DA UNIDADE

    ST_BT1 = 0;         //LIMPA FLAGS
    ST_BT2 = 0;
    F_FIM = 0;

    TEMPO = V_INICIO;                   //INICIA TEMPO
    LATD = CONVERTE[TEMPO];            //ATUALIZA O DISPLAY INICIALMENTE
    RCON = 0x7F;

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Rotina Principal                              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
    while(1)
    {
        FILTRO11 = 0;
        FILTRO21 = 0;
        FILTRO12 = T_FILTRO;
        FILTRO22 = T_FILTRO;
        LATD = CONVERTE[TEMPO];           //ATUALIZA O DISPLAY
        while (BT1)
        {
            ClrWdt();
            if (--FILTRO11 == 0)
            {
                FILTRO12--;
                if(FILTRO12 == 0)
                {
                    if (!ST_BT1)
                    {
                        LIGA_TIMER();
                        ST_BT1 = 1;
```

```
                    }
                }
            }
        }
        ST_BT1 = 0; // BOTÃO LIBERADO, LIMPA O FLAG
         while (BT2)
        {
            ClrWdt();
            if (--FILTRO21 == 0)
            {
                FILTRO22--;
                if(FILTRO22 == 0)
                {
                    if (!ST_BT2)
                    {
                        DESL_TIMER();
                        ST_BT2 = 1;
                    }
                }
            }
        }
        ST_BT2 = 0; // BOTÃO LIBERADO, LIMPA O FLAG
    }
}   // FIM DO PROGRAMA         ST_BT2 = 0; // BOTÃO LIBERADO, LIMPA O FLAG
```

7.6. Dicas e Comentários

No exemplo desta experiência, sempre que ocorre a interrupção, o valor do TMR0 é carregado com 131 de forma a obter uma contagem de 125 ciclos. Porém, o contador do TMR0 não ficou parado entre o instante em que a interrupção ocorreu e o instante em que ele foi novamente carregado. Por este motivo, pode ocorrer, que no instante em que o contador é novamente carregado com 131 ele já esteja com outro valor diferente de zero e aí a contagem não seria de exatamente 125 ciclos. Não é o caso deste exemplo, mais se o prescaler fosse de 1:1, ou seja, se o contador fosse incrementado a cada ciclo de máquina, quando a interrupção ocorresse e o contador fosse carregado, com certeza, o seu valor já seria diferente de zero. No caso do exemplo, o contador valeria 10, pois foram gastos 2 ciclos de máquina para desviar o programa para o vetor de interrupção, mais 3 ciclos para salvar o contexto, mais 2 ciclos para testar se a interrupção era de TMR0, mais 1 ciclo para limpar o bit da interrupção e mais 2 ciclos para efetivamente fazer a carga do contador. No total foram gastos 10 ciclos entre o instante em que a interrupção ocorreu e o instante em que o contador foi carregado com 131. Portanto, o tempo entre interrupções que deveria ser de 125 ciclos passou para 135 ciclos. Isto provoca um erro na contagem de tempo. Para corrigir este problema, deve-se somar o valor ao timer e não carrega-lo com um valor e desprezar o que ele já contou. Ainda admitindo que o prescaler fosse de 1:1, se fosse somado ao timer o valor 131, após a soma, o resultado final seria 141, sendo que 10 ele próprio já teria contado e 131 que foram somados. Desse instante até a próxima interrupção ocorreriam 115 contagens (256 – 141 = 115) e o tempo total entre interrupções se manteria conforme o desejado, ou seja, a cada 125 contagens.

7.7. Exercícios Propostos

1. Alterar o prescaler para 1:16 e obter a mesma base de tempo de 1 segundo.
2. Alterar o software para obter a mesma base de tempo de 1 segundo, porém sem utilizar o prescaler, ou seja, com prescaler configurado em 1:1.
3. Alterar a base de tempo do timer de 1 segundo para 60 segundos, ou seja, alterar o timer de segundos original para um timer de minutos.

8. Experiência 6 – Aceso à memória de dados EEPROM

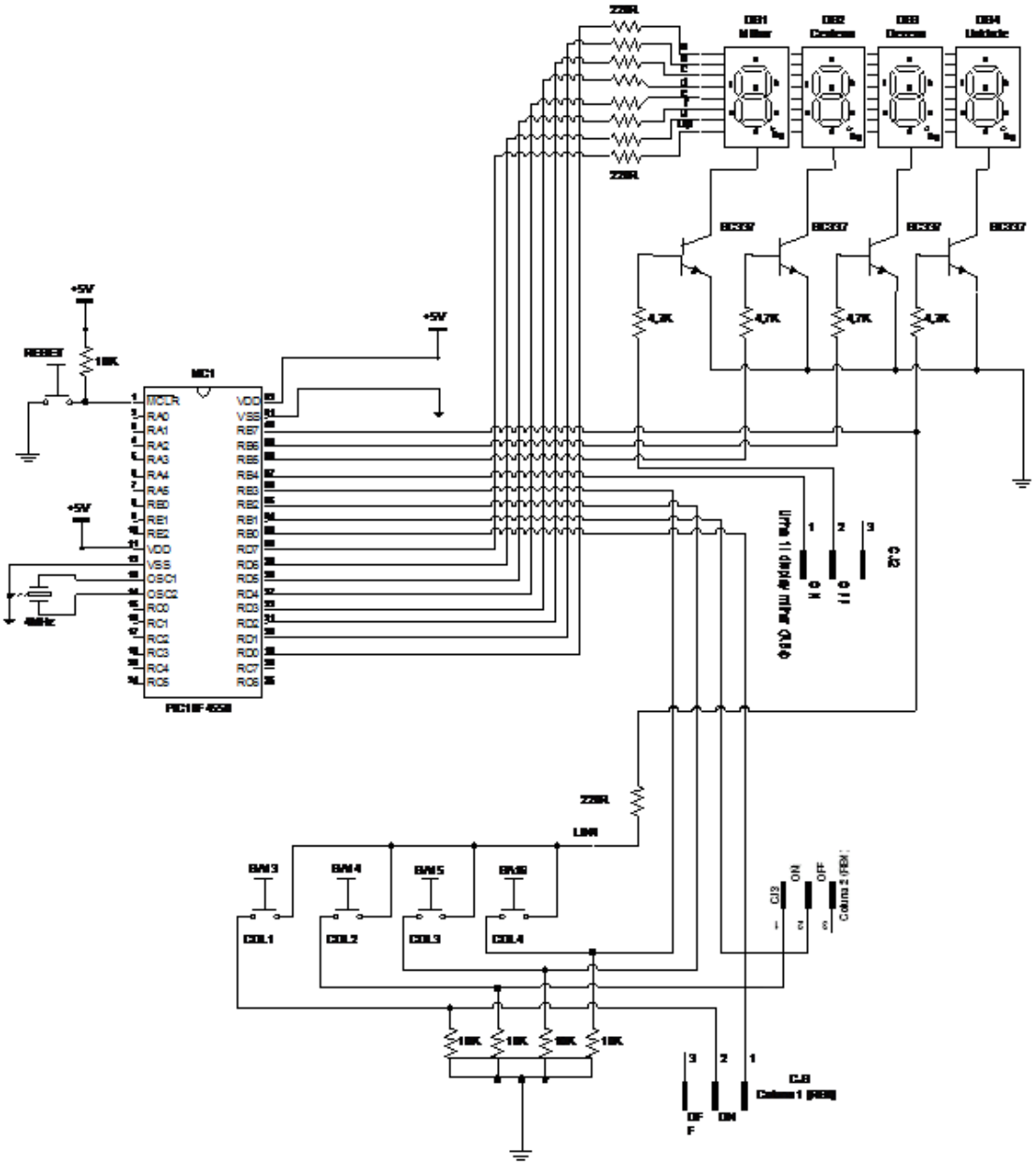
8.1. Objetivo

O objetivo desta experiência é o aprendizado da utilização da memória de dados não volátil EEPROM interna do microcontrolador.

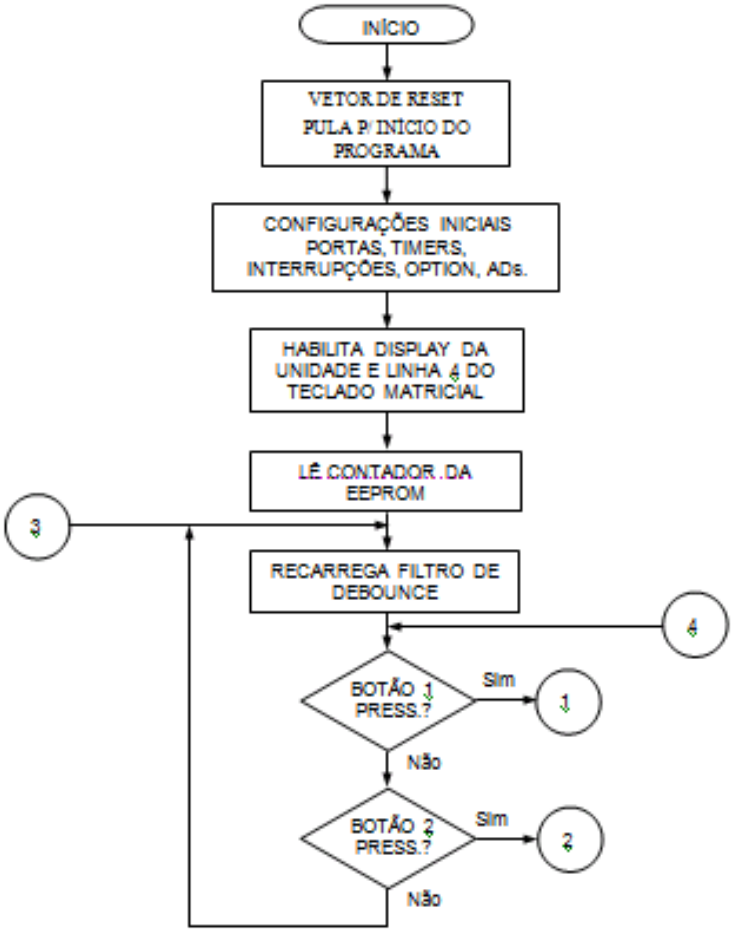
8.2. Descrição

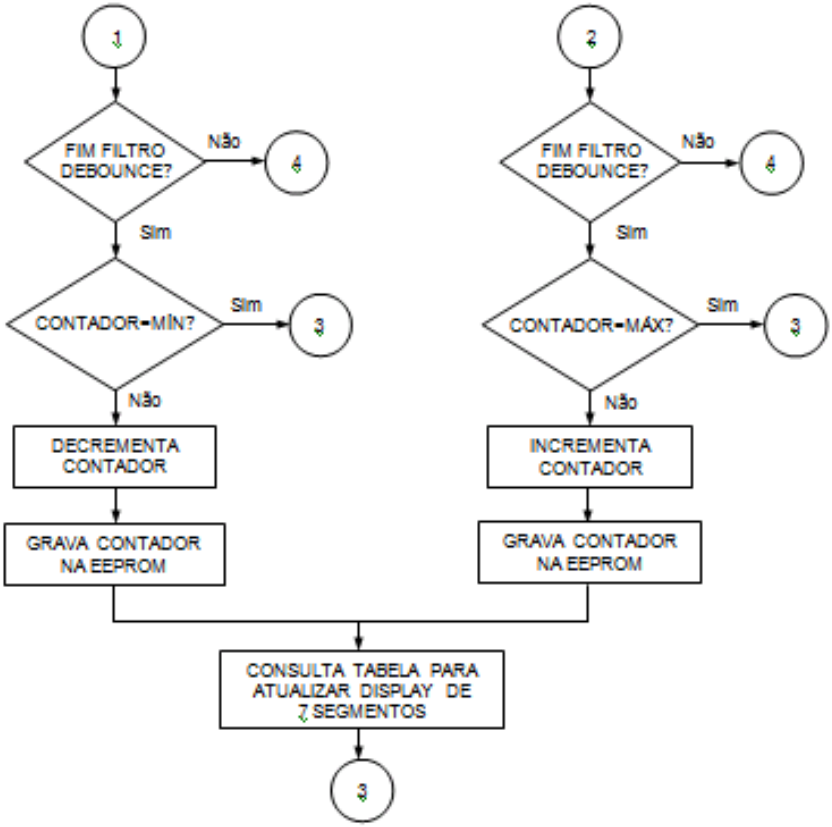
Nesta experiência foram implementadas rotinas para escrita e leitura de dados na memória EEPROM. O exemplo estudado na experiência 4 foi modificado e sempre que o valor da variável “CONTADOR” é alterado o seu valor é salvo na memória EEPROM. Ao inicializar o microcontrolador, a memória é lida e então o valor original pode ser restaurado. Assim, mesmo na queda de energia, o valor do contador é preservado.

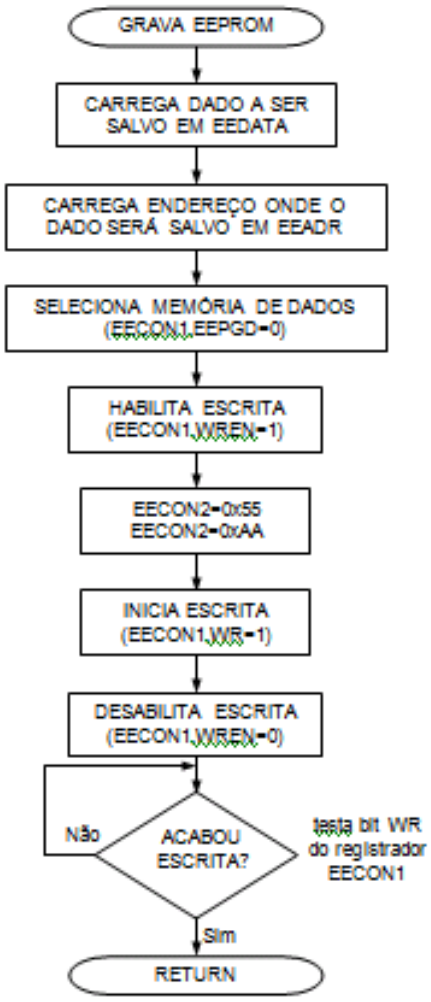
8.3. *Esquema Elétrico*

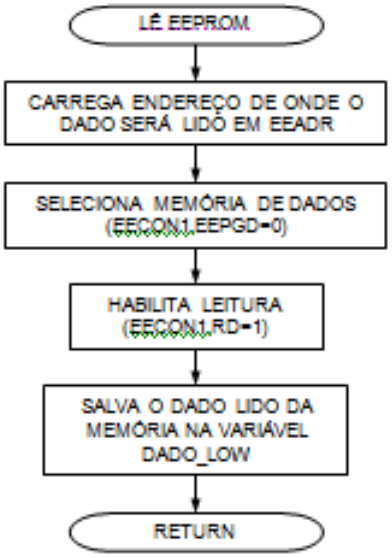


8.4. Fluxograma




1
FIM FILTRO DEBOUNCE?
Não
4
Sim
CONTADOR=MÍN?
Sim
3
Não
DECREMENTA CONTADOR
GRAVA CONTADOR NA EEPROM
2
FIM FILTRO DEBOUNCE?
Não
4
Sim
CONTADOR=MÁX?
Sim
3
Não
INCREMENTA CONTADOR
GRAVA CONTADOR NA EEPROM
CONSULTA TABELA PARA ATUALIZAR DISPLAY DE 7 SEGMENTOS
3


GRAVA EEPROM
CARREGA DADO A SER SALVO EM EEDATA
CARREGA ENDEREÇO ONDE O DADO SERÁ SALVO EM EEADR
SELECIONA MEMÓRIA DE DADOS (EECON1.EEPGD=0)
HABILITA ESCRITA (EECON1.WREN=1)
EECON2=0x55
EECON2=0xAA
INICIA ESCRITA (EECON1.WR=1)
DESABILITA ESCRITA (EECON1.WREN=0)
Não
ACABOU ESCRITA?
testa bit WR do registrador EECON1
Sim
RETURN


LÊ EEPROM
CARREGA ENDEREÇO DE ONDE O DADO SERÁ LIDO EM EEADR
SELECIONA MEMÓRIA DE DADOS (EECON1.EEPGD=0)
HABILITA LEITURA (EECON1.RD=1)
SALVA O DADO LIDO DA MEMÓRIA NA VARIÁVEL DADO_LOW
RETURN

8.5. Código

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Exemplo de Código para utilização com a McMaster           *
 *                              Exemplo 6                                  *
 *                                                                         *
 *                               MOSAICO                                   *
 *                                                                         *
 *  TEL: (0XX11) 4992-8775                SITE: www.mosaico.com.br         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                          *
 *   DATA : 18/01/2010                                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Descrição geral                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

//SOFTWARE QUE UTILIZA DOIS BOTÕES PARA INCREMENTAR E DECREMENTAR UM
//CONTADOR. O VALOR É MOSTRADO NO DISPLAY DE 7 SEGMENTOS.
//O VALOR É GRAVADO NA EEPROM INTERNA E AO LIGAR A PLACA O ÚLTIMO VALOR É
//MOSTRADO

/*
 CONFIGURAÇÂO DOS JUMPERS DA PLACA:

        CAP USB(RC3)               -> ON
        DATA I2C                   -> OFF
        CLOCK I2C                  -> OFF
        COLUNA 2(RB1)              -> ON
        COLUNA 1(RB0)              -> ON
        LINHA 1 / DISPLAY MILHAR(RB4)    -> ON
        LEDS ESPECIAS (RC0,RC1,RC2)      -> ON

*/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            DEFINIÇÃO PIC                                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <p18F4550.h>           // Register definitions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               INCLUDES DAS FUNÇÕES DE PERIFÉRICOS DO PIC                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <pwm.h>                //PWM library functions
#include <adc.h>                //ADC library functions
#include <timers.h>             //Timer library functions
#include <delays.h>             //Delay library functions
#include <i2c.h>                //I2C library functions
#include <stdlib.h>             //Library functions
#include <usart.h>              //USART library functions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      Configurações para gravação                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#pragma config FOSC = HS
#pragma config CPUDIV = OSC1_PLL2
#pragma config WDT = OFF
#pragma config WDTPS = 128
#pragma config LVP = OFF
#pragma config PWRT = ON
```

```
#pragma config BOR = OFF
#pragma config BORV = 0
#pragma config PBADEN = OFF
#pragma config DEBUG = OFF
#pragma config PLLDIV = 1
#pragma config USBDIV = 1
#pragma config FCMEN = OFF
#pragma config IESO = OFF
#pragma config VREGEN = OFF
#pragma config MCLRE = ON
#pragma config LPT1OSC = OFF
#pragma config CCP2MX = ON
#pragma config STVREN = OFF
#pragma config ICPRT = OFF

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Definição e inicialização das variáveis Globais          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.
unsigned char   CONTADOR;       //ARMAZENA O VALOR DA CONTAGEM
unsigned char   FILTRO11;       //FILTRAGEM  1 PARA O BOTÃO 1
unsigned char   FILTRO12;       //FILTRAGEM  2 PARA O BOTÃO 1
unsigned char   FILTRO21;       //FILTRAGEM  1 PARA O BOTÃO 2
unsigned char   FILTRO22;       //FILTRAGEM  2 PARA O BOTÃO 2

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                         Constantes internas                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de constantes facilita a programação e a manutenção.
#define     MIN         0
#define     MAX         15
#define     T_FILTRO    5

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Declaração dos flags de software                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de flags ajuda na programação e economiza memória RAM.
struct
{
    unsigned BIT0:1;
    unsigned BIT1:1;
    unsigned BIT2:1;
    unsigned BIT3:1;
    unsigned BIT4:1;
    unsigned BIT5:1;
    unsigned BIT6:1;
    unsigned BIT7:1;
}FLAGSbits;             //ARMAZENA OS FLAGS DE CONTROLE

#define     ST_BT1    FLAGSbits.BIT0        //STATUS DO BOTÃO 1
#define     ST_BT2    FLAGSbits.BIT1        //STATUS DO BOTÃO 2

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Definição e inicialização dos port's                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      PROTOTIPAGEM DE FUNÇÕES                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

 void write_EEPROM(unsigned char endereco, unsigned char dado);
 unsigned char read_EEPROM(unsigned char endereco);

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              ENTRADAS                                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
```

```
//As entradas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define     BT1     PORTBbits.RB0   //PORTA DO BOTÃO
                                    //1 -> PRESSIONADO
                                    //0 -> LIBERADO

#define     BT2     PORTBbits.RB1   //PORTA DO BOTÃO
                                    //1 -> PRESSIONADO
                                    //0 -> LIBERADO

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              SAÍDAS                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//As saídas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define     LINHA_4     PORTBbits.RB7   //PINO PARA ATIVAR LINHA 4 DO TECLADO
                                        //MATRICIAL
                                        //1 -> LINHA ATIVADA
                                        //0 -> LINHA DESATIVADA

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
// *                  ROTINA DE CONVERSÃO BINÁRIO -> DISPLAY              *
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
// ESTA ROTINA IRÁ RETORNAR EM W, O SIMBOLO CORRETO QUE DEVE SER
// MOSTRADO NO DISPLAY PARA CADA VALOR DE INTENSIDADE. O RETORNO JÁ ESTÁ
// FORMATADO PARA AS CONDIÇÕES DE LIGAÇÃO DO DISPLAY AO PORTD.
//          a
//     **********
//     *        *
//   f *        * b
//     *    g   *
//     **********
//     *        *
//   e *        * c
//     *    d   *
//     **********  *.

const rom unsigned char CONVERTE[16] =
{//   .GFEDCBA      POSIÇÃO CORRETA DOS SEGMENTOS
    0b00111111,     // 00 - RETORNA SÍMBOLO CORRETO 0
    0b00000110,     // 01 - RETORNA SÍMBOLO CORRETO 1
    0b01011011,     // 02 - RETORNA SÍMBOLO CORRETO 2
    0b01001111,     // 03 - RETORNA SÍMBOLO CORRETO 3
    0b01100110,     // 04 - RETORNA SÍMBOLO CORRETO 4
    0b01101101,     // 05 - RETORNA SÍMBOLO CORRETO 5
    0b01111101,     // 06 - RETORNA SÍMBOLO CORRETO 6
    0b00000111,     // 07 - RETORNA SÍMBOLO CORRETO 7
    0b01111111,     // 08 - RETORNA SÍMBOLO CORRETO 8
    0b01101111,     // 09 - RETORNA SÍMBOLO CORRETO 9
    0b01110111,     // 10 - RETORNA SÍMBOLO CORRETO A
    0b01111100,     // 11 - RETORNA SÍMBOLO CORRETO b
    0b00111001,     // 12 - RETORNA SÍMBOLO CORRETO C
    0b01011110,     // 13 - RETORNA SÍMBOLO CORRETO d
    0b01111001,     // 14 - RETORNA SÍMBOLO CORRETO E
    0b01110001};    // 15 - RETORNA SÍMBOLO CORRETO F

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Função de escrita da EEPROM interna                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void write_EEPROM(unsigned char endereco, unsigned char dado)
{
    EEDATA = dado;              //carrega dado
    EEADR = endereco;           //carrega endereço
```

```
    EECON1bits.CFGS = 0;          //habilita acesso a EEPROM
    EECON1bits.EEPGD = 0;         //aponta para memória de dados
    EECON1bits.WREN = 1;          //habilita escrita
    INTCONbits.GIE = 0;           //desabilita todas as interrupções
    EECON2 = 0x55;                //escreve 0x55 em EECON2 (obrigatório)
    EECON2 = 0xAA;                //escreve 0xAA em EECON2 (obrigatório)
    EECON1bits.WR = 1;            //inicia a escrita
    INTCONbits.GIE = 1;           //habilita todas as interrupções
    while(EECON1bits.WR);         //aguarda bit WR de EECON1 = 0
    EECON1bits.WREN = 0;          //desabilita escrita
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Função de leitura da EEPROM interna                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

unsigned char read_EEPROM(unsigned char endereco)
{
    EEADR = endereco;             //carrega endereço

    EECON1bits.CFGS = 0;          //habilita acesso a EEPROM
    EECON1bits.EEPGD = 0;         //aponta para memória de dados
    EECON1bits.RD = 1;            //habilita leitura

    return(EEDATA);               //retorna dado disponível em EEDATA
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função Principal                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void main ()
{
    PORTA = 0x00;                 //Limpa PORTA
    PORTB = 0x00;                 //Limpa PORTB
    PORTC = 0x00;                 //Limpa PORTC
    PORTD = 0x00;                 //Limpa PORTD
    PORTE = 0x00;                 //Limpa PORTE

    LATA = 0x00;                  //Limpa PORTA
    LATB = 0x00;                  //Limpa PORTB
    LATC = 0x00;                  //Limpa PORTC
    LATD = 0x00;                  //Limpa PORTD
    LATE = 0x00;                  //Limpa PORTE

    TRISA = 0b00101111;           //CONFIG DIREÇÃO DOS PINOS PORTA
    TRISB = 0b00001111;           //CONFIG DIREÇÃO DOS PINOS PORTB
    TRISC = 0b10011001;           //CONFIG DIREÇÃO DOS PINOS PORTC
    TRISD = 0b00000000;           //CONFIG DIREÇÃO DOS PINOS PORTD
    TRISE = 0b00000000;           //CONFIG DIREÇÃO DOS PINOS PORTE

    ADCON1 = 0b00001111;          //DESLIGA CONVERSORES A/D

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Inicialização do Sistema                         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

    LINHA_4 = 1;          //ATIVA A LINHA 4 DO TECLADO E DISPLAY DA UNIDADE

    ST_BT1 = 0;
    ST_BT2 = 0;

    CONTADOR = read_EEPROM(0X00) & 0x0F;

    PORTD = CONVERTE[CONTADOR];       //ATUALIZA O DISPLAY INICIALMENTE
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Rotina Principal                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
    while(1)
    {
        ClrWdt();
        FILTRO11 = 0;
        FILTRO21 = 0;
        FILTRO12 = T_FILTRO;
        FILTRO22 = T_FILTRO;

        while (BT1)
        {
            ClrWdt();
            if (--FILTRO11 == 0)
            {
                FILTRO12--;
                if(FILTRO12 == 0)
                {
                    if (CONTADOR != MIN)
                    {
                        if (!ST_BT1)
                        {
                            CONTADOR--;
                            PORTD = CONVERTE[CONTADOR];
                            write_EEPROM(0X00, CONTADOR);
                            ST_BT1 = 1;
                        }
                    }
                }
            }
        }
        ST_BT1 = 0; // BOTÃO LIBERADO, LIMPA O FLAG

        while (BT2)
        {
            ClrWdt();
            if (--FILTRO21 == 0)
            {
                FILTRO22--;
                if(FILTRO22 == 0)
                {
                    if (CONTADOR != MAX)
                    {
                        if (!ST_BT2)
                        {
                            CONTADOR++;
                            PORTD = CONVERTE[CONTADOR];
                            write_EEPROM(0X00, CONTADOR);
                            ST_BT2 = 1;
                        }
                    }
                }
            }
        }
        ST_BT2 = 0; // BOTÃO LIBERADO, LIMPA O FLAG
    }
}   // FIM DO PROGRAMA
```

8.6. Dicas e Comentários

A memória EEPROM do microcontrolador PIC18F4550 pode ser lida a qualquer momento e infinitas vezes, isto é, não existe um limite para a quantidade de leituras que são realizadas na memória de dados. Porém, no caso da escrita, a memória EEPROM do PIC18F4550 possui uma vida útil limitada. Atualmente o fabricante do componente garante uma vida útil de aproximadamente 1.000.000 de ciclos de escrita. Apesar deste número parecer extremamente alto, existem casos no qual este limite pode ser atingido com relativa facilidade. Portanto, sempre que se desenvolve um sistema que utilize a memória EEPROM e que necessite realizar um número de escritas considerável, deve-se pensar se o limite de 1.000.000 de ciclos será ou não atingido.

8.7. Exercícios Propostos

1. Alterar a posição onde o dado é armazenado na memória EEPROM.
2. Alterar o valor inicial do contador.
3. Acrescentar duas teclas, uma para salvar o dado na memória e outra para ler o dado da memória. Neste caso o contador não pode ser salvo sempre que alterado, apenas será salvo quando a tecla correta for pressionada. Quanto à leitura, deve ser realizada tanto na inicialização do software como quando forçada pela tecla.

9. Experiência 7 - Dimmer

9.1. Objetivo

O objetivo desta experiência é ensinar ao aluno como regular a potência fornecida a pequenas cargas através do microcontrolador. Para isso, é proposto um dimmer para uma lâmpada incandescente (leia obsevação abaixo) onde a potência é regulada através do uso de um PWM criado no software.

Obs: O acionamento da lâmpada é possível apenas quando trabalhamos com os microcontroladores PIC16F877A e PIC18F4520; o microcontrolador PIC18F4550 reserva esta saída para o módulo USB interno; neste caso o exemplo acionará um led especial associado ao pino RC0 do microcontrolador.

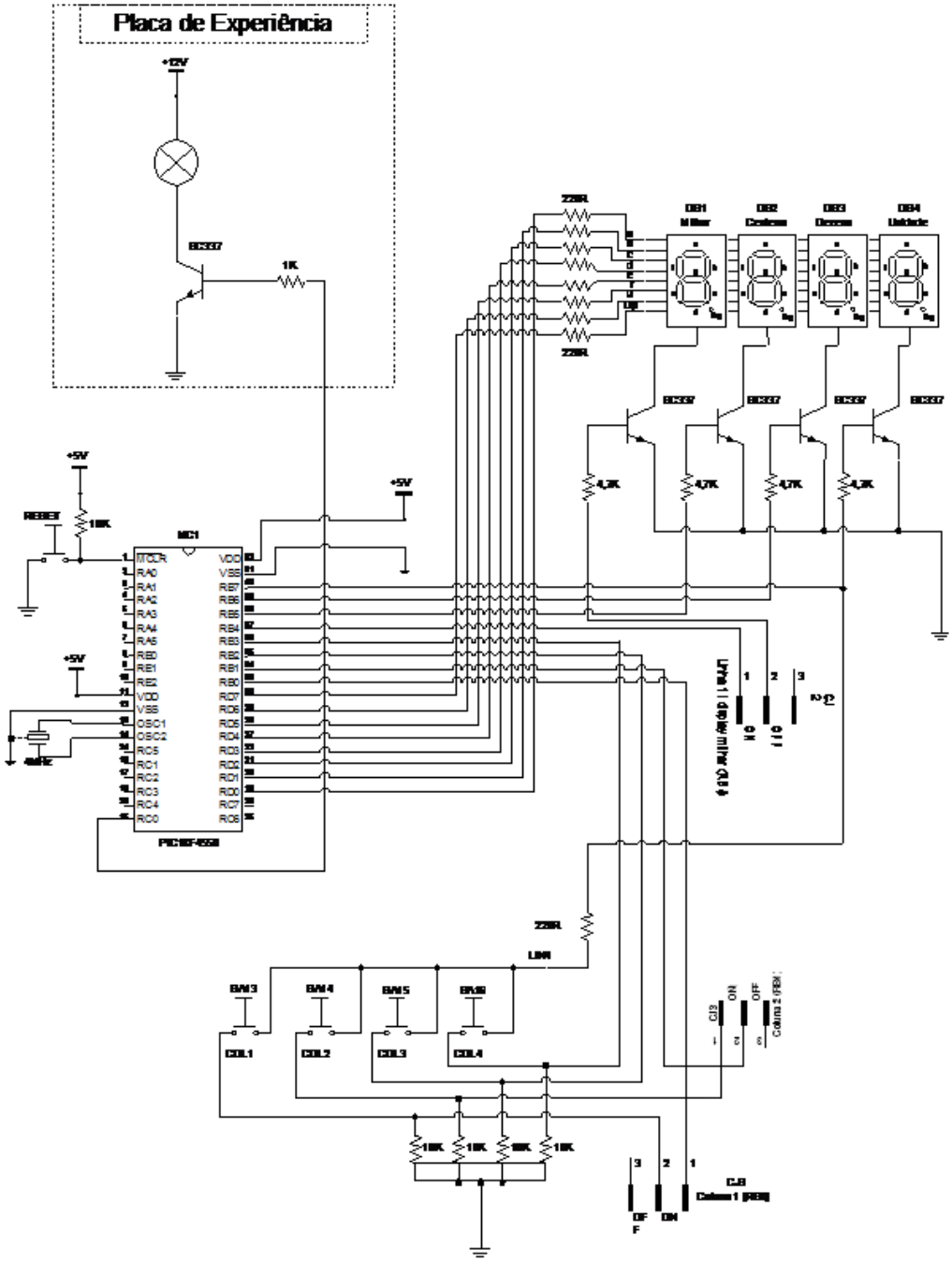
9.2. Descrição

Software que utiliza dois botões para incrementar e decrementar a intensidade do led. A interrupção de tmr0 é utilizada para controlar o PWM que aciona a lâmpada. A intensidade também é mostrada no display da unidade.

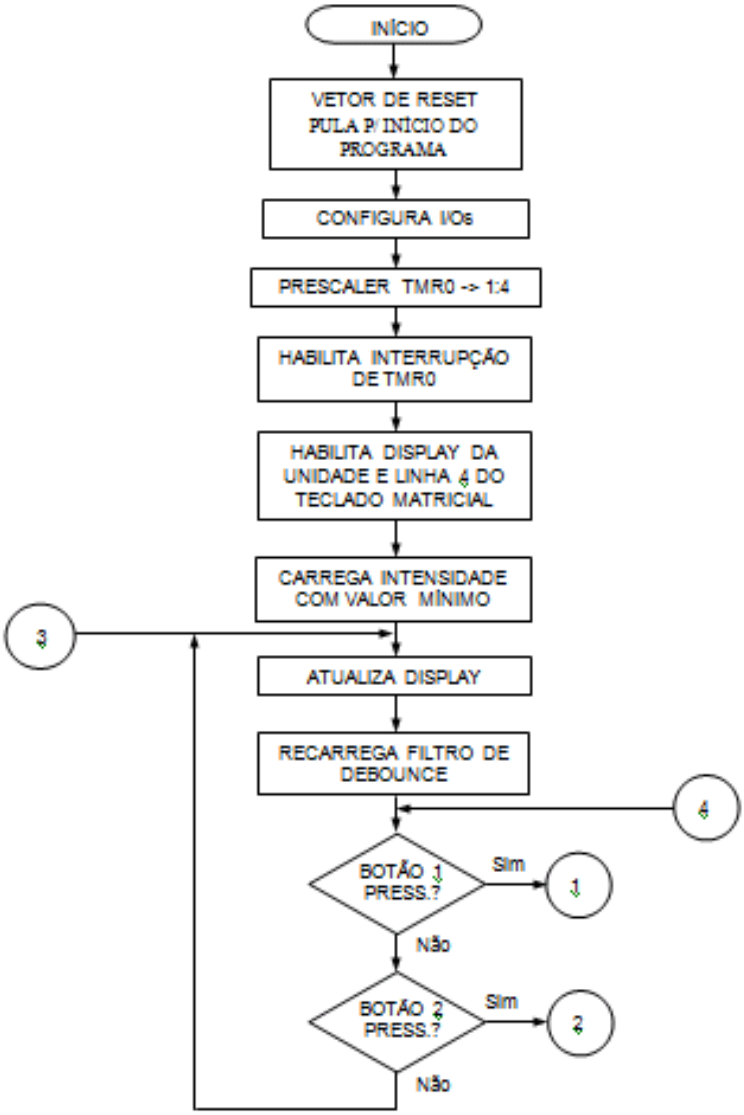
Os botões ativos são os da linha 4. O botão da coluna 1 incrementa a intensidade da lâmpada e o da coluna 2 decrementa a intensidade.

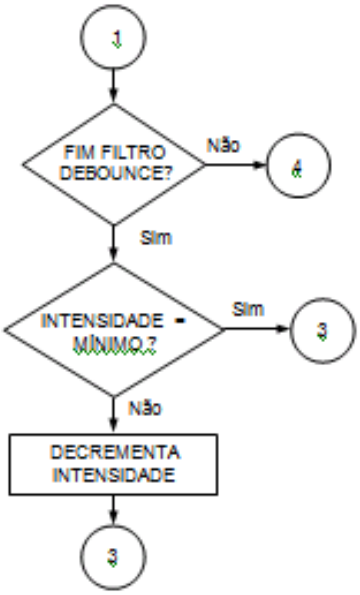
Foi criado um PWM com 4 bits de resolução de forma que existem 16 intensidades diferentes para a lâmpada. Para sua implementação, utilizou-se a interrupção de TMR0 sendo que a cada interrupção uma variável é incrementada. Esta variável é utilizada para subdividir o PWM em 16 intervalos. Comparando esta variável com o duty cycle ajustado pelo usuário, o software coloca o pino da lâmpada em nível lógico 1 ou 0, conforme a necessidade. A cada 16 interrupções o ciclo se repete e portanto o período do PWM é 16 vezes maior do que o período de uma interrupção. Este procedimento pode ser facilmente analisado pelo fluxograma da experiência.

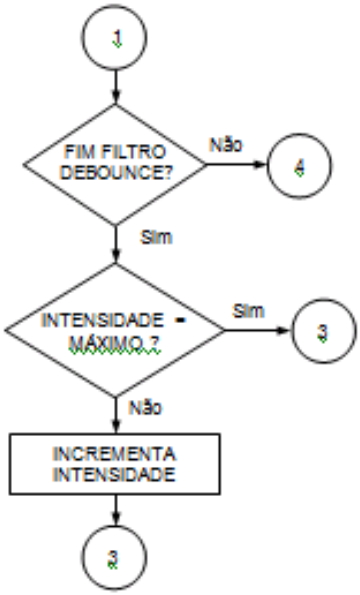
9.3. *Esquema Elétrico*

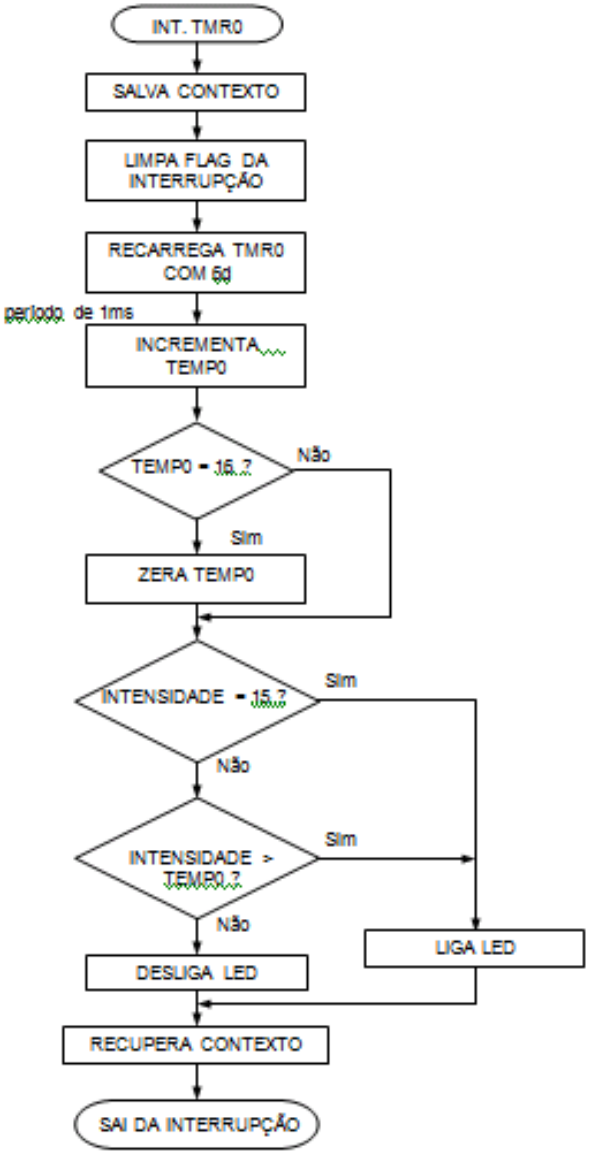


9.4. Fluxograma




1
FIM FILTRO DEBOUNCE?
Não
4
Sim
INTENSIDADE = MÍNIMO ?
Sim
3
Não
DECREMENTA INTENSIDADE
3


1
FIM FILTRO DEBOUNCE?
Não
4
Sim
INTENSIDADE = MÁXIMO ?
Sim
3
Não
INCREMENTA INTENSIDADE
3


INT. TMR0
SALVA CONTEXTO
LIMPA FLAG DA INTERRUPÇÃO
RECARREGA TMR0 COM 6d
período de 1ms
INCREMENTA TEMP0
TEMP0 = 16 ?
Não
Sim
ZERA TEMP0
INTENSIDADE = 15 ?
Sim
Não
INTENSIDADE > TEMP0 ?
Sim
Não
LIGA LED
DESLIGA LED
RECUPERA CONTEXTO
SAI DA INTERRUPÇÃO

9.5. Código

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Exemplo de Código para utilização com a McMaster            *
 *                              Exemplo 7                                  *
 *                                                                         *
 *                               MOSAICO                                   *
 *                                                                         *
 *  TEL: (0XX11) 4992-8775                SITE: www.mosaico.com.br         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                          *
 *   DATA : 18/01/2010                                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Descrição geral                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

//SOFTWARE QUE UTILIZA DOIS BOTÕES PARA INCREMENTAR E DECREMENTAR A
//INTENSIDADE DO LED. A INTERRUPÇÃO DE TMR0H É UTILIZADA PARA CONTROLAR
//O PWM QUE ACIONA O LED ESPECIAL (RC0) (CONFIGURAR O JUMPER "LEDS ESPECIAIS"
//NA POSIÇÃO ON. A INTENSIDADE TAMBÉM É MOSTRADA NO DISPLAY DA UNIDADE.
//OS BOTÕES ATIVOS SÃO O DA LINHA 4

/*
 CONFIGURAÇÂO DOS JUMPERS DA PLACA:

        CAP USB(RC3)                -> ON
        DATA I2C                    -> OFF
        CLOCK I2C                   -> OFF
        COLUNA 2(RB1)               -> ON
        COLUNA 1(RB0)               -> ON
        LINHA 1 / DISPLAY MILHAR(RB4)    -> ON
        LEDS ESPECIAS (RC0,RC1,RC2)      -> ON

*/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            DEFINIÇÃO PIC                                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <p18F4550.h>           // Register definitions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               INCLUDES DAS FUNÇÕES DE PERIFÉRICOS DO PIC                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <pwm.h>                //PWM library functions
#include <adc.h>                //ADC library functions
#include <timers.h>             //Timer library functions
#include <delays.h>             //Delay library functions
#include <i2c.h>                //I2C library functions
#include <stdlib.h>             //Library functions
#include <usart.h>              //USART library functions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      Configurações para gravação                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#pragma config FOSC = HS
#pragma config CPUDIV = OSC1_PLL2
#pragma config WDT = OFF
#pragma config WDTPS = 128
#pragma config LVP = OFF
```

```
#pragma config PWRT = ON
#pragma config BOR = OFF
#pragma config BORV = 0
#pragma config PBADEN       = OFF
#pragma config DEBUG        = OFF
#pragma config PLLDIV       = 1
#pragma config USBDIV       = 1
#pragma config FCMEN        = OFF
#pragma config IESO  = OFF
#pragma config VREGEN       = OFF
#pragma config MCLRE        = ON
#pragma config LPT1OSC      = OFF
#pragma config CCP2MX = ON
#pragma config STVREN       = OFF
#pragma config ICPRT        = OFF


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Definição e inicialização das variáveis Globais          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.
unsigned char   INTENSIDADE;    //ARMAZENA O VALOR DA CONTAGEM
unsigned char   FILTRO11;       //FILTRAGEM  1 PARA O BOTÃO 1
unsigned char   FILTRO12;       //FILTRAGEM  2 PARA O BOTÃO 1
unsigned char   FILTRO21;       //FILTRAGEM  1 PARA O BOTÃO 2
unsigned char   FILTRO22;       //FILTRAGEM  2 PARA O BOTÃO 2
unsigned char   TEMPO;          //INTERVALOS DE 1 MS

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Constantes internas                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de constantes facilita a programação e a manutenção.
#define     MIN         0
#define     MAX         15
#define     T_FILTRO    5

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Declaração dos flags de software                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de flags ajuda na programação e economiza memória RAM.
struct
{
    unsigned BIT0:1;
    unsigned BIT1:1;
    unsigned BIT2:1;
    unsigned BIT3:1;
    unsigned BIT4:1;
    unsigned BIT5:1;
    unsigned BIT6:1;
    unsigned BIT7:1;
}FLAGSbits;             //ARMAZENA OS FLAGS DE CONTROLE

#define    ST_BT1    FLAGSbits.BIT0        //STATUS DO BOTÃO 1
#define    ST_BT2    FLAGSbits.BIT1        //STATUS DO BOTÃO 2

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Definição e inicialização dos port's                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      PROTOTIPAGEM DE FUNÇÕES                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void TRATA_INT_TIMER0 (void);

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
 *                                ENTRADAS                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//As entradas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define     BT1     PORTBbits.RB0   //PORTA DO BOTÃO
                                    //1 -> PRESSIONADO
                                    //0 -> LIBERADO

#define     BT2     PORTBbits.RB1   //PORTA DO BOTÃO
                                    //1 -> PRESSIONADO
                                    //0 -> LIBERADO

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                               SAÍDAS                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//As saídas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define     LINHA_4     LATBbits.LATB7   //PINO PARA ATIVAR LINHA 4 DO TECLADO
                                        //MATRICIAL
                                        //1 -> LINHA ATIVADA
                                        //0 -> LINHA DESATIVADA

#define     LAMPADA     LATCbits.LATC0  //DEFINE PINO DA LAMPADA
                                        //0 -> LÂMPADA APAGADA
                                        //1 -> LÂMPADA ACESA

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
// *                 ROTINA DE TRATAMENTO DE INT DE TIMER0                  *
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
#pragma code TIMER0_interrupt = 0x0008
void TIMER0_int (void)
{
  _asm goto TRATA_INT_TIMER0 _endasm
}
#pragma code

#pragma interrupt TRATA_INT_TIMER0
void TRATA_INT_TIMER0 (void)
{
    INTCONbits.TMR0IF = 0;                          //LIMPA FLAG DE INTERRUPÇÃO
    WriteTimer0(256 - 250);
    TEMPO++;
    if (TEMPO == 16)TEMPO = 0;
    if (TEMPO >= INTENSIDADE) LAMPADA = 0;
    else LAMPADA = 1;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
// *                   ROTINA DE CONVERSÃO BINÁRIO -> DISPLAY               *
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
// ESTA ROTINA IRÁ RETORNAR EM W, O SIMBOLO CORRETO QUE DEVE SER
// MOSTRADO NO DISPLAY PARA CADA VALOR DE INTENSIDADE. O RETORNO JÁ ESTÁ
// FORMATADO PARA AS CONDIÇÕES DE LIGAÇÃO DO DISPLAY AO PORTD.
//          a
//     **********
//     *        *
//   f *        * b
//     *    g   *
//     **********
//     *        *
//   e *        * c
//     *    d   *
//     **********  *.
```

```
const rom unsigned char CONVERTE[16] =
{//   .GFEDCBA       POSIÇÃO CORRETA DOS SEGMENTOS
    0b00111111,     // 00 - RETORNA SÍMBOLO CORRETO 0
    0b00000110,     // 01 - RETORNA SÍMBOLO CORRETO 1
    0b01011011,     // 02 - RETORNA SÍMBOLO CORRETO 2
    0b01001111,     // 03 - RETORNA SÍMBOLO CORRETO 3
    0b01100110,     // 04 - RETORNA SÍMBOLO CORRETO 4
    0b01101101,     // 05 - RETORNA SÍMBOLO CORRETO 5
    0b01111101,     // 06 - RETORNA SÍMBOLO CORRETO 6
    0b00000111,     // 07 - RETORNA SÍMBOLO CORRETO 7
    0b01111111,     // 08 - RETORNA SÍMBOLO CORRETO 8
    0b01101111,     // 09 - RETORNA SÍMBOLO CORRETO 9
    0b01110111,     // 10 - RETORNA SÍMBOLO CORRETO A
    0b01111100,     // 11 - RETORNA SÍMBOLO CORRETO b
    0b00111001,     // 12 - RETORNA SÍMBOLO CORRETO C
    0b01011110,     // 13 - RETORNA SÍMBOLO CORRETO d
    0b01111001,     // 14 - RETORNA SÍMBOLO CORRETO E
    0b01110001};    // 15 - RETORNA SÍMBOLO CORRETO F

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função Principal                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void main ()
{
    PORTA = 0x00;               //Limpa PORTA
    PORTB = 0x00;               //Limpa PORTB
    PORTC = 0x00;               //Limpa PORTC
    PORTD = 0x00;               //Limpa PORTD
    PORTE = 0x00;               //Limpa PORTE

    LATA = 0x00;                //Limpa PORTA
    LATB = 0x00;                //Limpa PORTB
    LATC = 0x00;                //Limpa PORTC
    LATD = 0x00;                //Limpa PORTD
    LATE = 0x00;                //Limpa PORTE

    TRISA = 0b00101111;         //CONFIG DIREÇÃO DOS PINOS PORTA
    TRISB = 0b00001111;         //CONFIG DIREÇÃO DOS PINOS PORTB
    TRISC = 0b10011000;         //CONFIG DIREÇÃO DOS PINOS PORTC
    TRISD = 0b00000000;         //CONFIG DIREÇÃO DOS PINOS PORTD
    TRISE = 0b00000000;         //CONFIG DIREÇÃO DOS PINOS PORTE

    ADCON1 = 0b00001111;        //DESLIGA CONVERSORES A/D

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Inicialização do Sistema                      *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
    OpenTimer0(TIMER_INT_ON & T0_8BIT & T0_SOURCE_INT & T0_PS_1_4);

    LINHA_4 = 1;        //ATIVA A LINHA 4 DO TECLADO E DISPLAY DA UNIDADE

    ST_BT1 = 0;
    ST_BT2 = 0;
    INTENSIDADE = MIN;                  //INICIA INTENSIDADE = MIN
    WriteTimer0(256 - 250);             //REINICIA TMR0H, SETA TIMER P/ 1ms
    PORTD = CONVERTE[INTENSIDADE];      //ATUALIZA O DISPLAY INICIALMENTE
    RCON = 0x00;

    INTCONbits.GIEH = 1;

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Rotina Principal                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
    while(1)
    {
```

```
        ClrWdt();
        FILTRO11 = 0;
        FILTRO21 = 0;
        FILTRO12 = T_FILTRO;
        FILTRO22 = T_FILTRO;

        while (BT1)
        {
            ClrWdt();
            if (--FILTRO11 == 0)
            {
                FILTRO12--;
                if(FILTRO12 == 0)
                {
                    if (INTENSIDADE != MIN)
                    {
                        if (!ST_BT1)
                        {
                            INTENSIDADE--;
                            PORTD = CONVERTE[INTENSIDADE];
                            ST_BT1 = 1;
                        }
                    }
                }
            }
        }
        ST_BT1 = 0; // BOTÃO LIBERADO, LIMPA O FLAG

        while (BT2)
        {
            ClrWdt();
            if (--FILTRO21 == 0)
            {
                FILTRO22--;
                if(FILTRO22 == 0)
                {
                    if (INTENSIDADE != MAX)
                    {
                        if (!ST_BT2)
                        {
                            INTENSIDADE++;
                            PORTD = CONVERTE[INTENSIDADE];
                            ST_BT2 = 1;
                        }
                    }
                }
            }
        }
        ST_BT2 = 0; // BOTÃO LIBERADO, LIMPA O FLAG
    }
}   // FIM DO PROGRAMA
```

9.6. Dicas e Comentários

A técnica utilizada nesta experiência para implementar o PWM apresenta como inconveniente resultar em PWMs de baixa freqüência. Por exemplo, no caso desta experiência a interrupção ocorre a cada 1ms e como o PWM têm 4 bits de resolução, temos um período total de 16ms para o PWM. Isto significa que a freqüência é de 62,5Hz, ou seja, o PWM implementado via software apresenta baixa freqüência. Além disso, se ele fosse de por exemplo 8 bits, o período passaria para 256ms e, portanto, a freqüência cairia para aproximadamente 4Hz, ou seja, extremamente baixa. Note que quanto maior a resolução desejada maior é o período do PWM.

9.7. Exercícios Propostos

1. Altere a freqüência do PWM alterando a base de tempo do TMR0 para 100us.
2. Altere o PWM para trabalhar com apenas 2 bits de resolução.
3. Acrescente as rotinas de leitura e escrita na EEPROM para salvar e recuperar a intensidade da lâmpada mesmo na falta de energia.

10. Experiência 8 – Botões, Leds e Buzzer

10.1. Objetivo

O objetivo desta experiência é dar continuidade ao aprendizado das interrupções e em particular à interrupção de TMR2 utilizada nesta experiência para excitar o buzzer.

10.2. Descrição

O buzzer utilizado no McMaster não é do tipo auto-oscilante, ou seja, este tipo de buzzer precisa ser excitado externamente para emitir algum som. Assim, o microcontrolador deve ser responsável por gerar uma onda quadrada a fim de excitar o piezo-elétrico e fazer com que o buzzer emita algum som. Operar com este tipo de buzzer é muito mais trabalhoso do que com um buzzer auto-oscilante, já que não basta ligar-lo, é necessário aplicar uma onda quadrada para que o buzzer funcione. Porém, ele dá margem à criação de diferentes tons, pois, quem gera a freqüência da onda quadrada e conseqüentemente o tom com que o buzzer irá tocar é o microcontrolador e se este gerar freqüências adequadas, pode-se implementar até mesmo uma simples melodia.

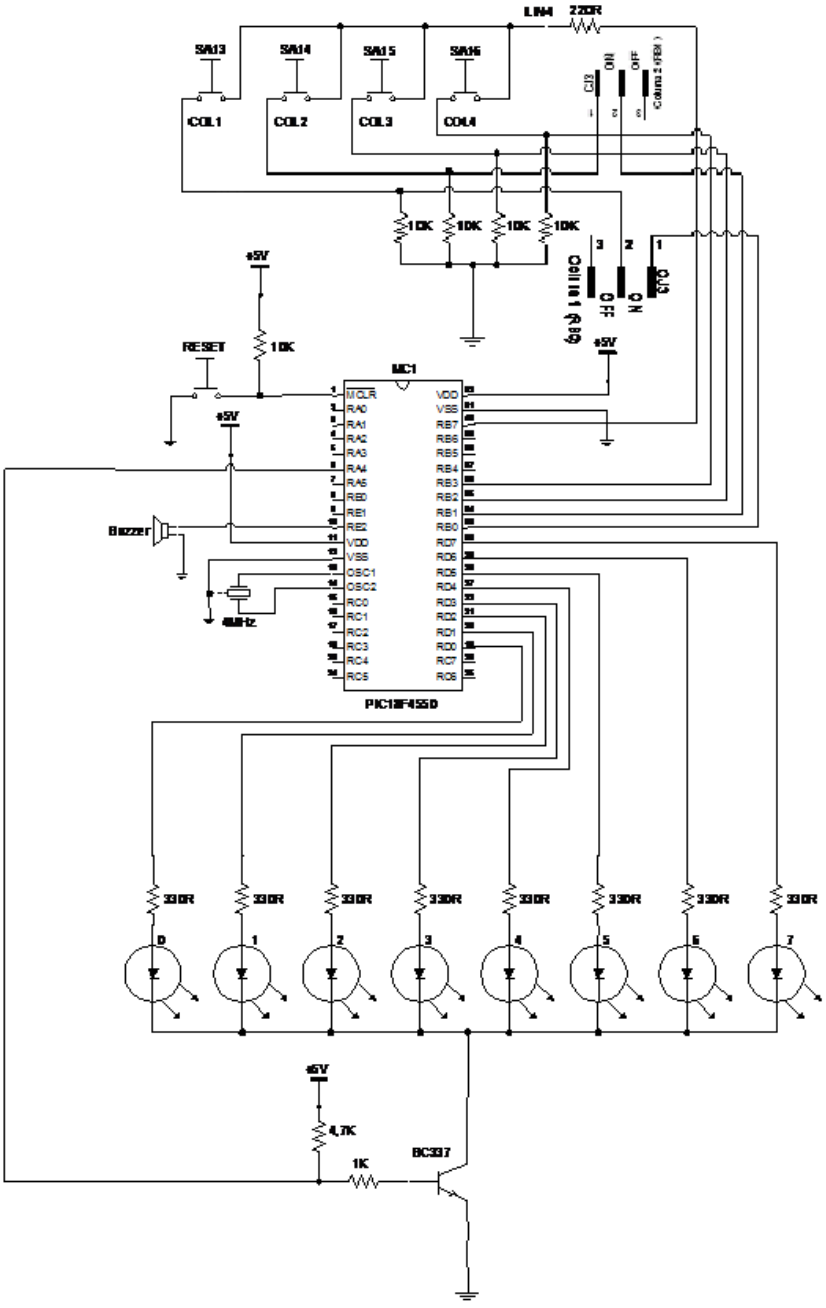
Apesar desta experiência utilizar o TMR2 para excitar o buzzer com freqüência variável, o objetivo não é a implementação de notas musicais, embora isto seja possível.

Diferente dos outros timers do microcontrolador, o TMR2 estoura e pode gerar uma interrupção sempre que o seu valor ultrapassa o valor de um registrador especial, denominado **PR2**. Ou seja, o TMR2, diferentemente dos outros timers conta desde zero até o valor programado no registrador PR2 e não até 255 (a não ser que o valor do registrador PR2 seja 255).

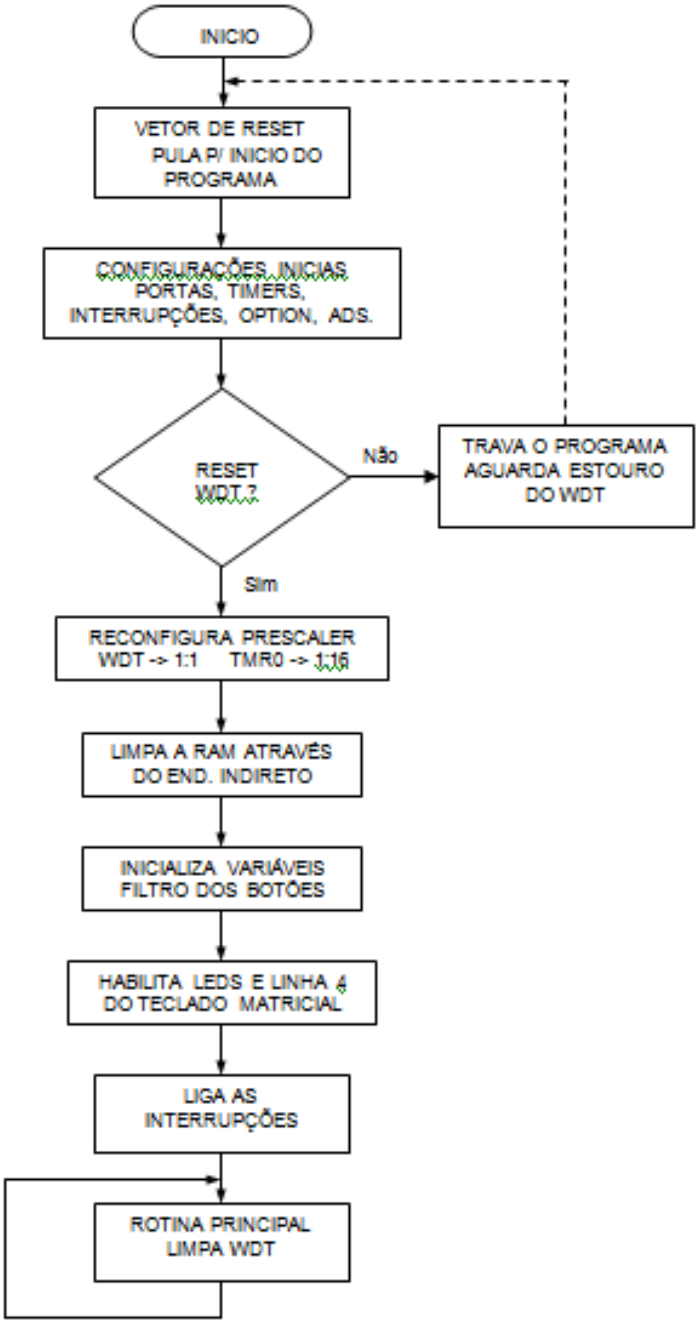
Desta forma, o que o software faz é ler o estado das teclas da linha 4 e em função deste alterar o valor do registrador **PR2**. Conseqüentemente, a interrupção de TMR2 varia de período, ou seja, a freqüência com que a interrupção ocorre depende do estado das teclas da linha 4. Como é a interrupção a responsável por gerar a onda quadrada que excita o buzzer e como a freqüência pode ser alterada pelas teclas, dependendo da configuração de teclas pressionadas o buzzer emite som em diferentes tonalidades, criando, embora com freqüência erradas uma espécie de mini-teclado.

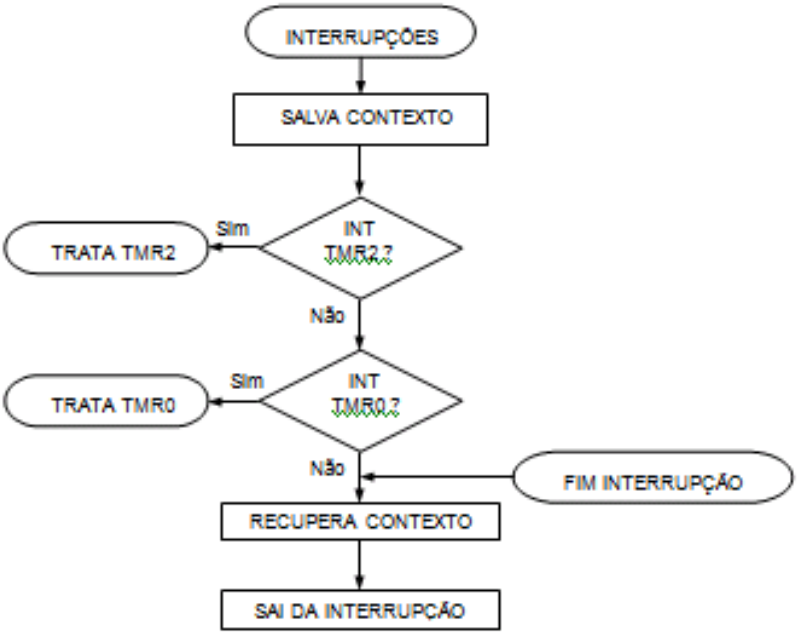
Os leds ligados ao PORTD são utilizados apenas para repetir o estado das teclas pressionadas a fim de criar um efeito visual para a experiência.

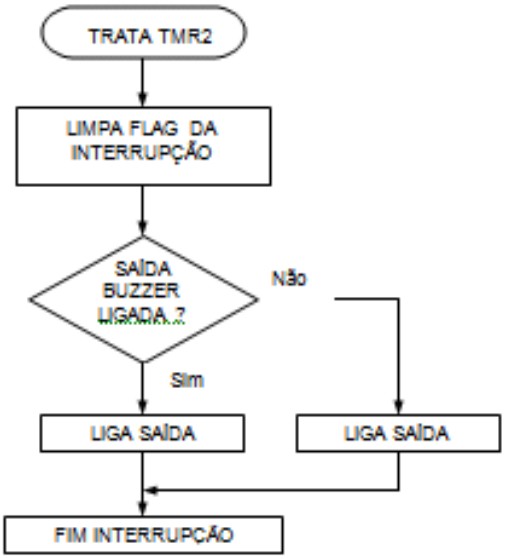
10.3. Esquema Elétrico

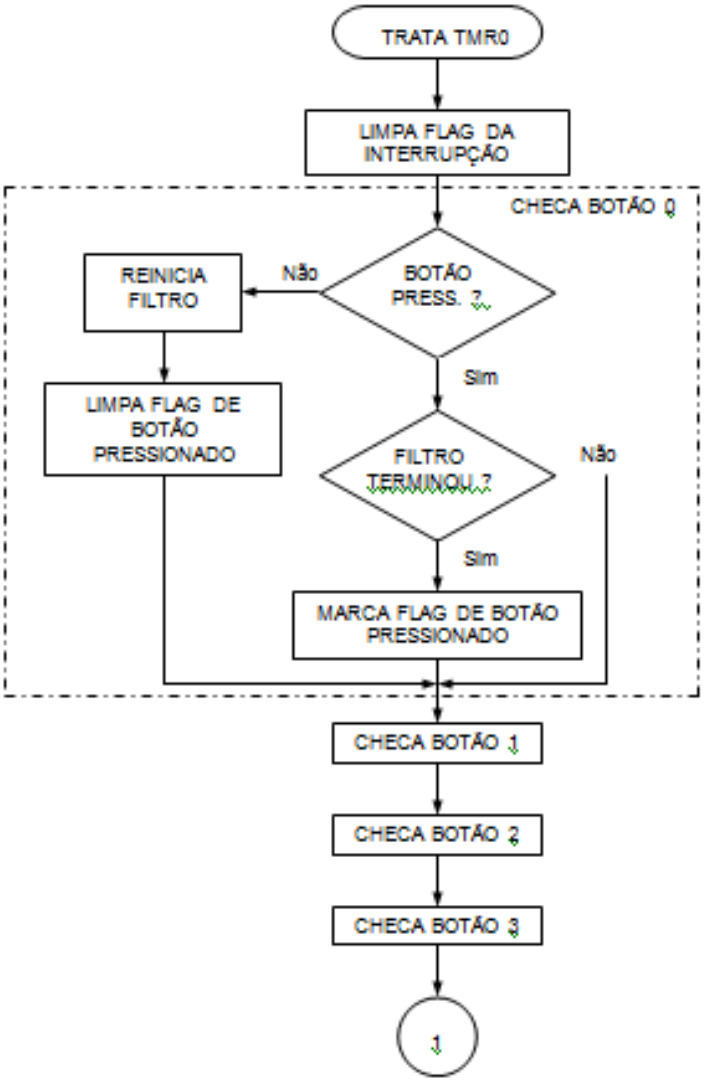


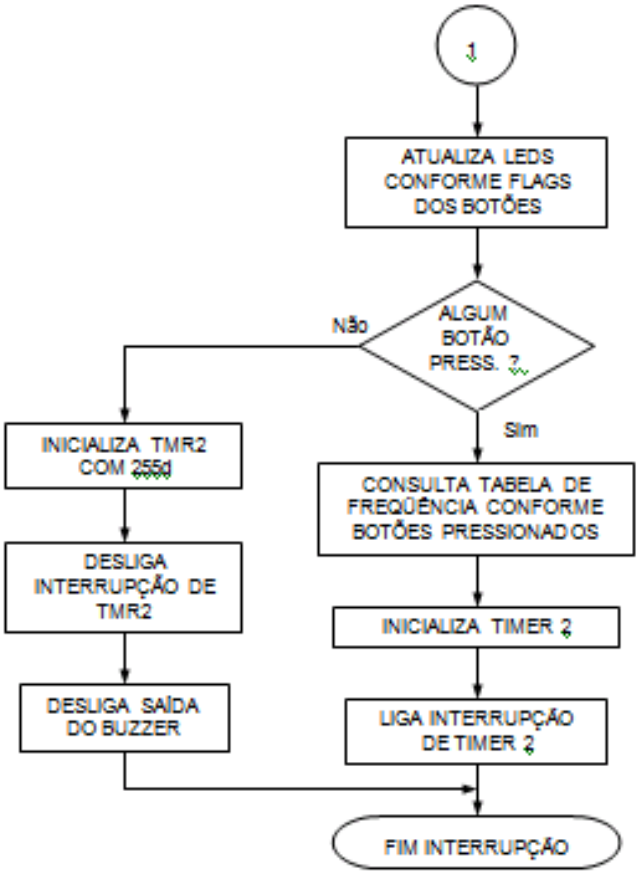
10.4. Fluxograma




INTERRUPÇÕES
SALVA CONTEXTO
INT TMR2 ?
Sim
TRATA TMR2
Não
INT TMR0 ?
Sim
TRATA TMR0
Não
FIM INTERRUPÇÃO
RECUPERA CONTEXTO
SAI DA INTERRUPÇÃO


TRATA TMR2
LIMPA FLAG DA INTERRUPÇÃO
SAÍDA BUZZER LIGADA ?
Não
Sim
LIGA SAÍDA
LIGA SAÍDA
FIM INTERRUPÇÃO


TRATA TMR0
LIMPA FLAG DA INTERRUPÇÃO
CHECA BOTÃO 0
BOTÃO PRESS. ?
Não
REINICIA FILTRO
Sim
LIMPA FLAG DE BOTÃO PRESSIONADO
FILTRO TERMINOU ?
Não
Sim
MARCA FLAG DE BOTÃO PRESSIONADO
CHECA BOTÃO 1
CHECA BOTÃO 2
CHECA BOTÃO 3
1


1
ATUALIZA LEDS CONFORME FLAGS DOS BOTÕES
ALGUM BOTÃO PRESS. ?
Não
Sim
INICIALIZA TMR2 COM 255d
DESLIGA INTERRUPÇÃO DE TMR2
DESLIGA SAÍDA DO BUZZER
CONSULTA TABELA DE FREQÜÊNCIA CONFORME BOTÕES PRESSIONADOS
INICIALIZA TIMER 2
LIGA INTERRUPÇÃO DE TIMER 2
FIM INTERRUPÇÃO

10.5. Código

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Exemplo de Código para utilização com a McMaster            *
 *                              Exemplo 8                                  *
 *                                                                         *
 *                               MOSAICO                                   *
 *                                                                         *
 *  TEL: (0XX11) 4992-8775                SITE: www.mosaico.com.br         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                          *
 *   DATA : 18/01/2010                                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Descrição geral                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

//ESTE SOFTWARE ESTÁ PREPARADO PARA LER QUATRO BOTÕES E TOCAR O BUZZER
//COM DURAÇÃO VARIÁVEL CONFORME TECLA PRESSIONADA.

/*
 CONFIGURAÇÂO DOS JUMPERS DA PLACA:

        CAP USB(RC3)               -> ON
        DATA I2C                   -> OFF
        CLOCK I2C                  -> OFF
        COLUNA 2(RB1)              -> ON
        COLUNA 1(RB0)              -> ON
        LINHA 1 / DISPLAY MILHAR(RB4)    -> ON
        LEDS ESPECIAS (RC0,RC1,RC2)      -> ON

*/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           DEFINIÇÃO PIC                                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <p18F4550.h>           // Register definitions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               INCLUDES DAS FUNÇÕES DE PERIFÉRICOS DO PIC                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <pwm.h>                //PWM library functions
#include <adc.h>                //ADC library functions
#include <timers.h>             //Timer library functions
#include <delays.h>             //Delay library functions
#include <i2c.h>                //I2C library functions
#include <stdlib.h>             //Library functions
#include <usart.h>              //USART library functions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Configurações para gravação                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#pragma config FOSC = HS
#pragma config CPUDIV = OSC1_PLL2
#pragma config WDT = OFF
#pragma config WDTPS = 128
#pragma config LVP = OFF
#pragma config PWRT = ON
#pragma config BOR = OFF
#pragma config BORV = 0
```

```
#pragma config PBADEN = OFF
#pragma config DEBUG = OFF
#pragma config PLLDIV = 1
#pragma config USBDIV = 1
#pragma config FCMEN = OFF
#pragma config IESO = OFF
#pragma config VREGEN = OFF
#pragma config MCLRE = ON
#pragma config LPT1OSC = OFF
#pragma config CCP2MX = ON
#pragma config STVREN = OFF
#pragma config ICPRT = OFF

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Definição e inicialização das variáveis Globais          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.
unsigned char filtro1;
unsigned char filtro2;
unsigned char filtro3;
unsigned char filtro4;
unsigned char status_leds=0;            // atualiza leds conforme o botão pressionado

unsigned char tabela[16] = {255,16,32,48,64,80,96,112,128,144,160,176,192,208,224,240};

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                         Constantes internas                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de constantes facilita a programação e a manutenção.
#define     MIN         0
#define     MAX         15
#define     t_filtro    25

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Declaração dos flags de software                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de flags ajuda na programação e economiza memória RAM.
//Este programa não utiliza nenhum flag de usuário

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Definição e inicialização dos port's                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       PROTOTIPAGEM DE FUNÇÕES                         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void TRATA_HIGH_INT(void);
void TRATA_INT_TIMER0(void);
void TRATA_INT_TIMER2(void);

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              ENTRADAS                                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// As entradas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define     botao1  PORTBbits.RB0   //PORTA DO BOTÃO
                                    //1 -> PRESSIONADO
                                    //0 -> LIBERADO

#define     botao2  PORTBbits.RB1   //PORTA DO BOTÃO
                                    //1 -> PRESSIONADO
                                    //0 -> LIBERADO

#define     botao3  PORTBbits.RB2   //PORTA DO BOTÃO
```

```
                                    //1 -> PRESSIONADO
                                    //0 -> LIBERADO

#define     botao4  PORTBbits.RB3   //PORTA DO BOTÃO
                                    //1 -> PRESSIONADO
                                    //0 -> LIBERADO

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                                SAÍDAS                                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//As saídas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define    C_LEDS  LATAbits.LATA4   //PINO PARA ATIVAR GRUPO DE 8 LEDS
                                    //1 -> LEDS ATIVADOS
                                    //0 -> LEDS DESATIVADOS

#define    LINHA_4 LATBbits.LATB7   //PINO PARA ATIVAR LINHA 4 DO TECLADO
                                    //MATRICIAL
                                    //1 -> LINHA ATIVADA
                                    //0 -> LINHA DESATIVADA

#define    buzzer  LATEbits.LATE2

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
// *             ROTINA DE TRATAMENTO DE INT DE ALTA PRIORIDADE            *
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
#pragma code VETOR_HIGH_PRIORITY = 0x0008
void HIGH_int (void)
{
  _asm goto TRATA_HIGH_INT _endasm
}
#pragma code

#pragma interrupt TRATA_HIGH_INT
void TRATA_HIGH_INT(void)
{
    if(PIR1bits.TMR2IF)      TRATA_INT_TIMER2();
    if(INTCONbits.TMR0IF)    TRATA_INT_TIMER0();
}

void TRATA_INT_TIMER2(void)
{
    PIR1bits.TMR2IF = 0;     //LIMPA FLAG DE INTERRUPÇÃO

    if (buzzer != 0)         // o buzzer está ligado ?
    {
        buzzer = 0;          // sim, então desliga
    }
    else
    {
        buzzer = 1;          // não, então liga
    }
}

void TRATA_INT_TIMER0(void)
{
    INTCONbits.TMR0IF = 0;   //LIMPA FLAG DE INTERRUPÇÃO

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Tratamento do Botão 1                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

    if (botao1)              // testa botão 1
    {                        // botão 1 está pressionado ?
        filtro1--;           // Sim, então decrementa o filtro
```

```
        if (filtro1 == 0)     // acabou o filtro do botão 1 ?
        {
            status_leds = status_leds | 0b00000001;    // marca que o botão está
pressionado

        }

    }
    else
    {                         // botão 1 liberado
        filtro1 = t_filtro;  // inicia o filtro do botão 1
        status_leds = status_leds & 0b11111110;        // marca que o botão foi
liberado
    }

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Tratamento do Botão 2                            *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

    if (botao2)               // testa botão 2
    {                         // botão 2 está pressionado ?
        filtro2--;            // Sim, então decrementa o filtro
        if (filtro2 == 0)     // fim do filtro do botão 2 ?
        {
            status_leds = status_leds | 0b00000010;    // marca que o botão está
pressionado
        }

    }
    else
    {                         // botão 2 liberado
        filtro2 = t_filtro;  // inicia o filtro do botão 2
        status_leds = status_leds & 0b11111101;        // marca que o botão foi
liberado
    }

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Tratamento do Botão 3                            *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

    if (botao3)               // testa botão 3
    {                         // botão 3 está pressionado ?
        filtro3--;            // Sim, então decrementa o filtro
        if (filtro3 == 0)     // fim do filtro do botão 3 ?
        {
            status_leds = status_leds | 0b00000100;    // marca que o botão está
pressionado
        }

    }
    else
    {                         // botão 3 liberado
        filtro3 = t_filtro;  // inicia o filtro do botão 3
        status_leds = status_leds & 0b11111011;         // marca que o botão foi
liberado
    }

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Tratamento do Botão 4                            *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

    if (botao4)               // testa botão 4
    {                         // botão 4 está pressionado ?
        filtro4--;            // Sim, então decrementa o filtro
        if (filtro4 == 0)     // fim do filtro do botão 4 ?
        {
```

```
            status_leds = status_leds | 0b00001000;     // marca que o botão está
pressionado
        }

    }
    else
    {                           // botão 4 liberado
        filtro4 = t_filtro;  // inicia o filtro do botão 4
        status_leds = status_leds & 0b11110111;         // marca que o botão foi
liberado
    }

    PORTD = status_leds;


    if (status_leds == 0)
    {
        PR2 = 0xFF;
        CloseTimer2();
        buzzer = 0;
    }
    else
    {
        PR2 = tabela[status_leds];    // consulta tabela e inicializa timer2.
        OpenTimer2(TIMER_INT_ON & T2_PS_1_1 & T2_POST_1_3);

    }
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função Principal                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void main ()
{
    PORTA = 0x00;               //Limpa PORTA
    PORTB = 0x00;               //Limpa PORTB
    PORTC = 0x00;               //Limpa PORTC
    PORTD = 0x00;               //Limpa PORTD
    PORTE = 0x00;               //Limpa PORTE

    LATA = 0x00;                //Limpa PORTA
    LATB = 0x00;                //Limpa PORTB
    LATC = 0x00;                //Limpa PORTC
    LATD = 0x00;                //Limpa PORTD
    LATE = 0x00;                //Limpa PORTE

    TRISA = 0b00101111;         //CONFIG DIREÇÃO DOS PINOS PORTA
    TRISB = 0b00001111;         //CONFIG DIREÇÃO DOS PINOS PORTB
    TRISC = 0b10011111;         //CONFIG DIREÇÃO DOS PINOS PORTC
    TRISD = 0b00000000;         //CONFIG DIREÇÃO DOS PINOS PORTD
    TRISE = 0b00000011;         //CONFIG DIREÇÃO DOS PINOS PORTE

    ADCON1 = 0b11111111;        //DESLIGA CONVERSORES A/D

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Inicialização do Sistema                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
    OpenTimer0(TIMER_INT_ON & T0_8BIT & T0_SOURCE_INT & T0_PS_1_4);

    PR2 = 0xff;
    OpenTimer2(TIMER_INT_OFF & T2_PS_1_1 & T2_POST_1_3);

    RCON = 0x00;

    C_LEDS = 1;                //ATIVA LEDS LIGADOS AO PORTD
```

```
    LINHA_4 = 1;                //ATIVA LINHA 4 DO TECLADO MATRICIAL

    status_leds = 0;

    filtro1 = t_filtro;         // inicia filtro do botao1
    filtro2 = t_filtro;         // inicia filtro do botao2
    filtro3 = t_filtro;         // inicia filtro do botao3
    filtro4 = t_filtro;         // inicia filtro do botao4

        INTCONbits.TMR0IE = 1;           // liga chave da int. do timer 0
  PIE1bits.TMR2IE     = 1;          // liga chave da int. do timer 2
  INTCONbits.PEIE = 1;              // liga chave de periféricos
  INTCONbits.GIE = 1;               // liga chave geral

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Rotina Principal                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
    while(1)
    {
        Nop();
    }
}                               // FIM DO PROGRAMA
```

10.6. Dicas e Comentários

Notar que nesta experiência foi utilizada uma tabela para retornar em função do estado das teclas pressionadas o valor que deve ser carregado no registrador PR2. Ou seja, conforme comentado na experiência 4, a utilização de tabelas não se limita apenas à conversão de BCD para display de 7 segmentos. Prova disso é que nesta experiência a tabela foi útil para exercer função completamente diferente.

O software desta experiência apresenta uma particularidade. Veja que não existe nenhuma instrução, a não ser o CLRWDT, no loop principal do programa. Assim, todas as instruções do software e toda a sua lógica está implementada apenas dentro das interrupções, sendo parte na interrupção de TMR0 e parte na interrupção de TMR2.

10.7. Exercícios Propostos

1. Alterar a configuração do TMR2 para que o buzzer seja excitado em outra faixa de freqüências.
2. Utilizar a instrução boleana XOR para inverter o estado do pino do buzzer.
3. Inverter a escala de freqüências, trocando a mais alta pela mais baixa.

11. Experiência 9 – Varredura de displays e utilização do TIMER 1

11.1. Objetivo

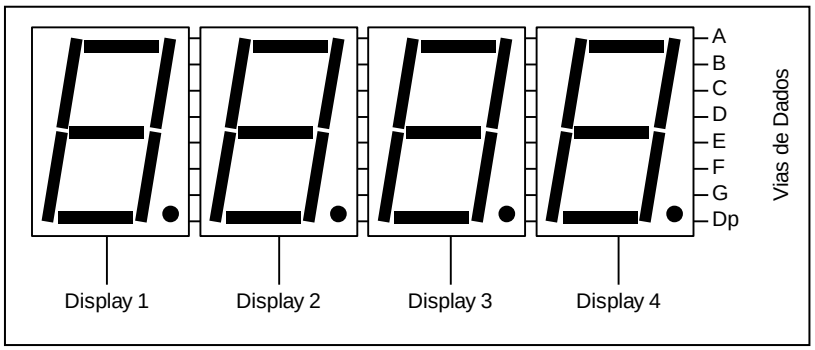
O objetivo desta experiência é o aprendizado do sistema de varredura comumente utilizado para varrer displays de 7 segmentos. Além disso, é visto o TMR1 fechando assim o estudo sobre os três times do PIC18F4550.

11.2. Descrição

Cada display é formado por um conjunto de 8 leds, sendo 7 para a caracterização do dígito e 1 para o ponto decimal. Desta forma, precisaremos de um pino do PIC para controlar cada um dos leds e, portanto, serão necessários 8 pinos para acionar os 7 segmentos e mais o ponto decimal. Porém, o McMaster não possui apenas um display e sim quatro. Seriam necessários então 32 pinos do microcontrolador para acionar os quatro displays? Não, existe uma saída para isso. O segredo para minimizar a quantidade de pinos utilizados é o emprego de um conceito denominado varredura.

Para isso, interligam-se todos os displays, juntando todos os pinos de um mesmo segmento numa única via, de forma a criar um barramento de dados com as vias de A até Dp. Em seguida, utiliza-se um pino para controlar o comum de cada um dos displays (total de 4 pinos). Assim, quando se deseja escrever em um dos displays, bastar informar os segmentos a serem acionados nas vias de dados e ligar o comum do display desejado.

Utilizando o hardware desta forma é fácil notar que apenas um dos displays poderá ser acionado de cada vez. Porém, se acionarmos os displays continuamente, um após o outro e de forma rápida, nossos olhos não conseguiram perceber que apenas um display está acionado por vez dando a impressão de que todos os displays estão acionados o tempo todo.



Assim, empregando-se a técnica de varredura, consegue-se controlar os 4 displays utilizando apenas 12 pinos do microcontrolador. O exemplo desenvolvido para esta experiência faz muito mais que simplesmente implementar a varredura dos displays. Trata-se de um contador regressivo de segundos, ou seja, um temporizador capaz de contar até 9.999 segundos.

As teclas habilitadas são as da linha 4 e seguem as funções descritas na tabela.

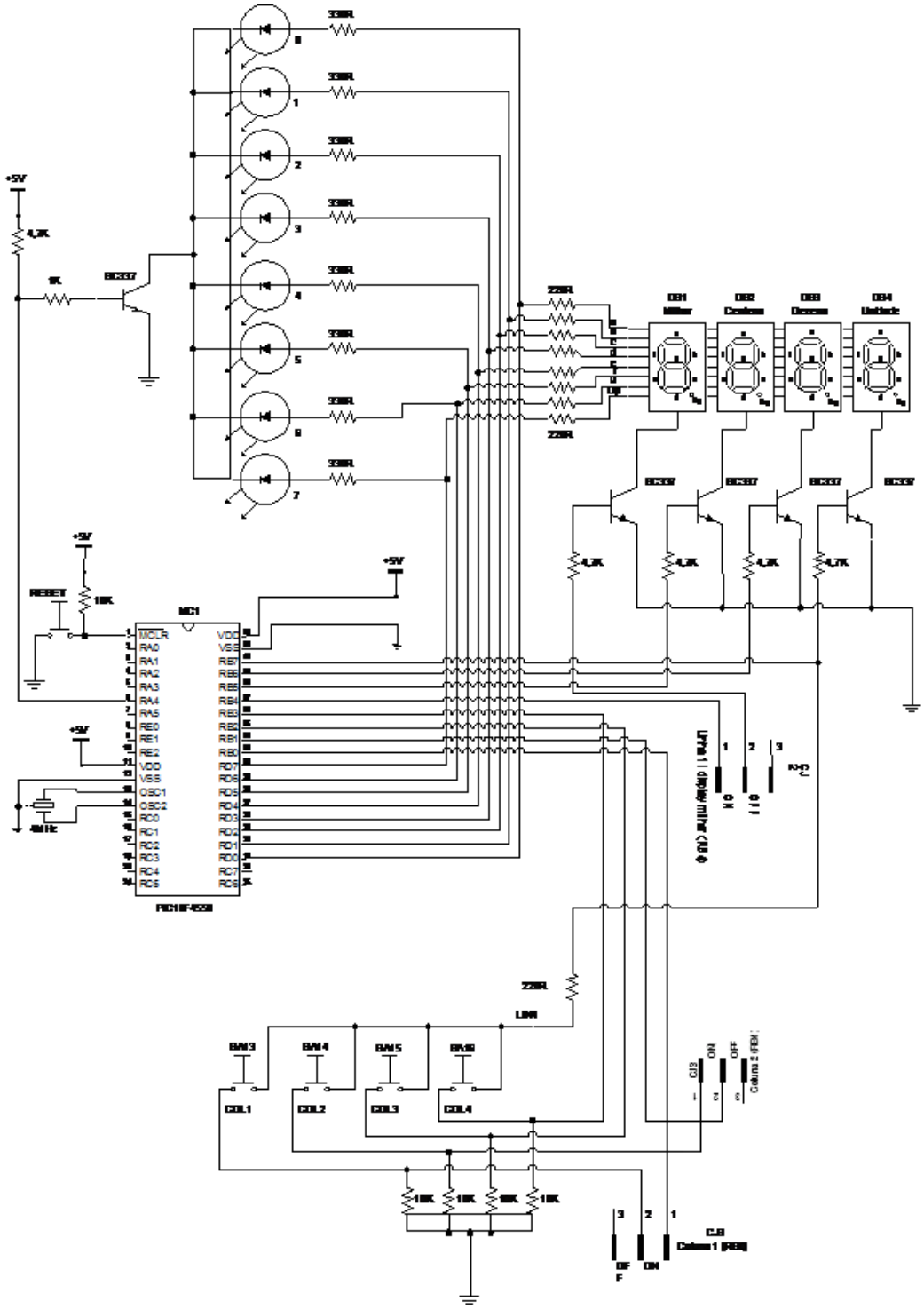
Coluna	Descrição
1	nenhuma função
2	Incrementa o valor inicial em 1 segundo
3	Decrementa o valor inicial em 1 segundo
4	Inicia e paralisa o temporizador

Para a contagem do tempo utilizou-se a interrupção de TMR1, configurada conforme a tabela a seguir.

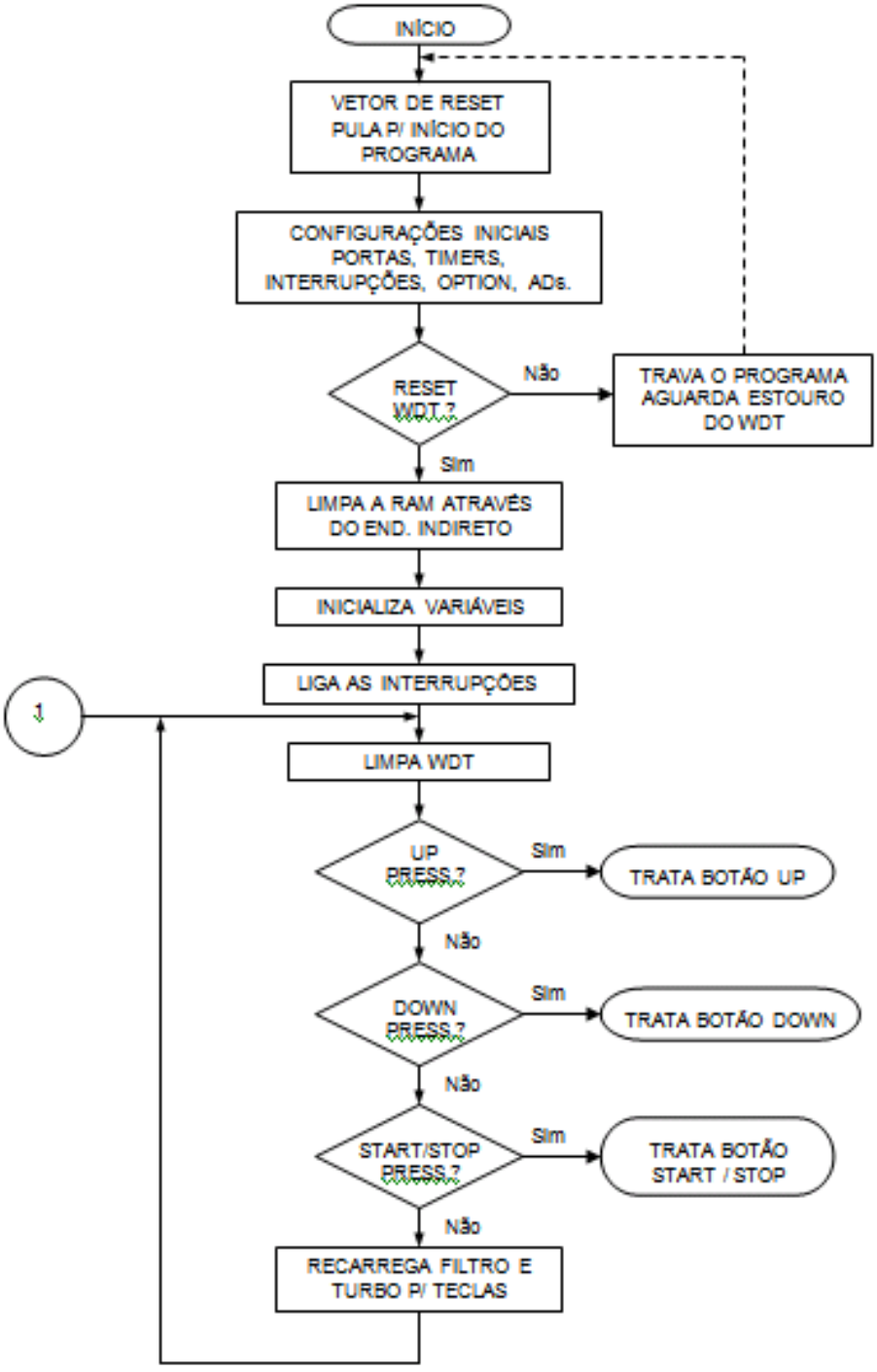
Ciclo de Maq.	*Prescale*	Conta TMR1	Auxiliar	Período	Freqüência
1 µs	8	62500	2	1.000.000 µs	1 Hz

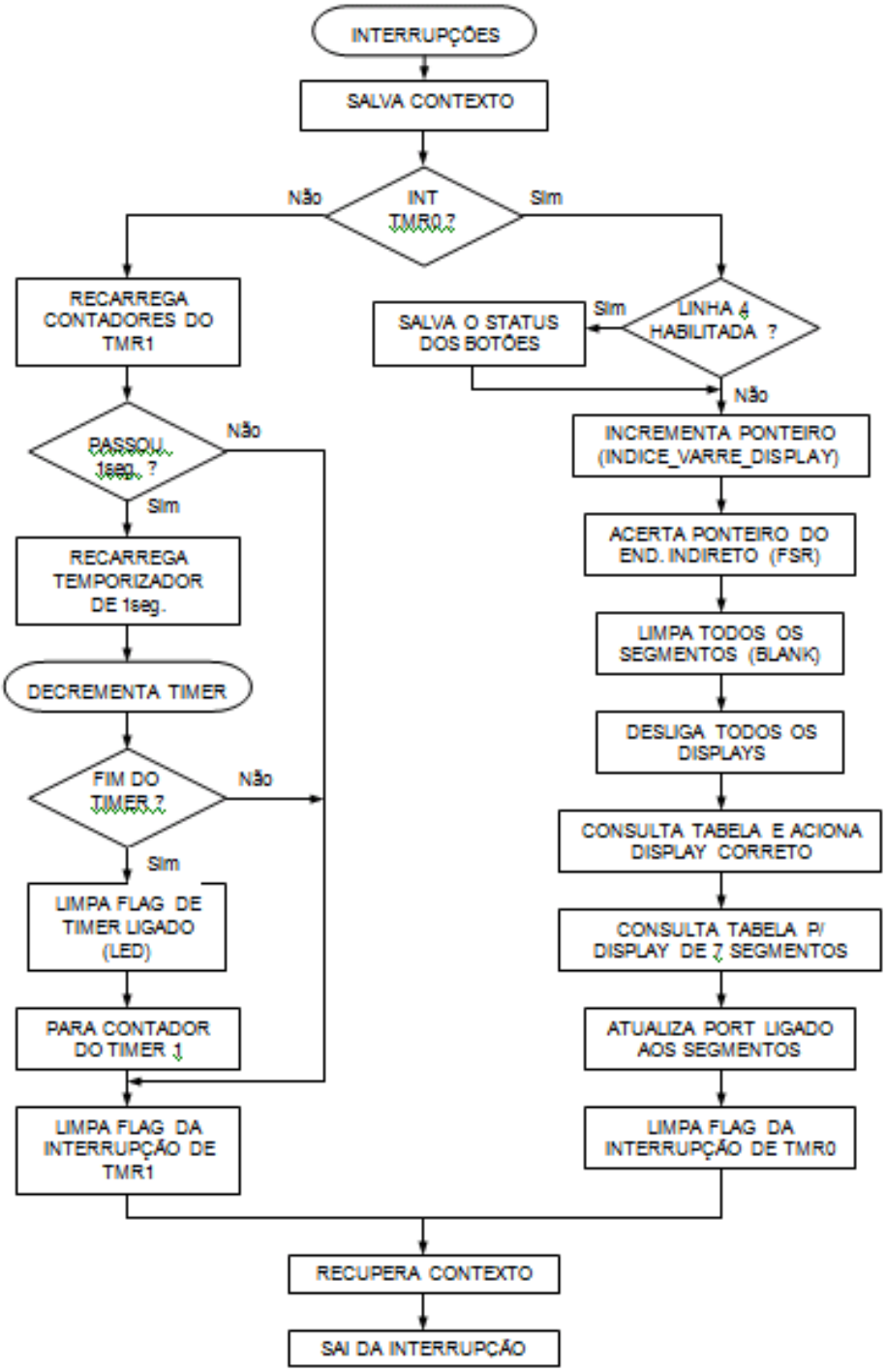
Configurou-se o *prescale* do TMR1 em 1:8 e o contador foi inicializado com o valor total menos o desejado para a contagem (65.536 – 62.500). Desta maneira a interrupção ocorre a cada 0,5 segundo. A fim de criar a base de tempo de 1 segundo foi utilizada uma variável auxiliar que é decrementada a cada ocorrência da interrupção.

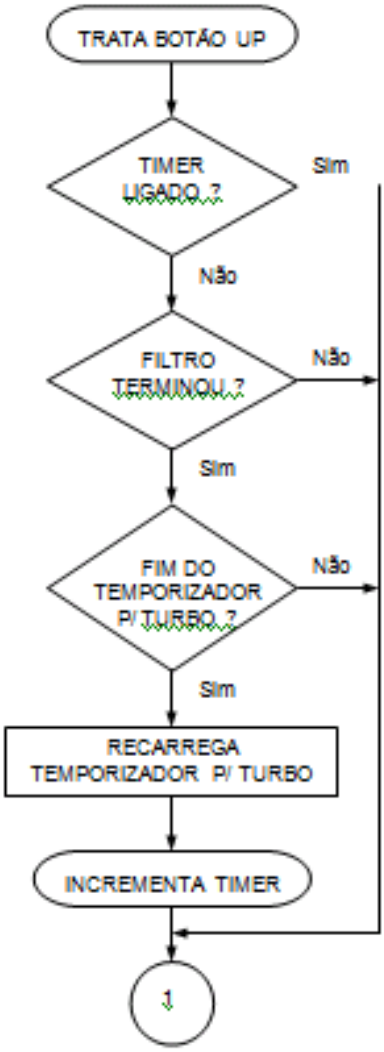
11.3. Esquema Elétrico

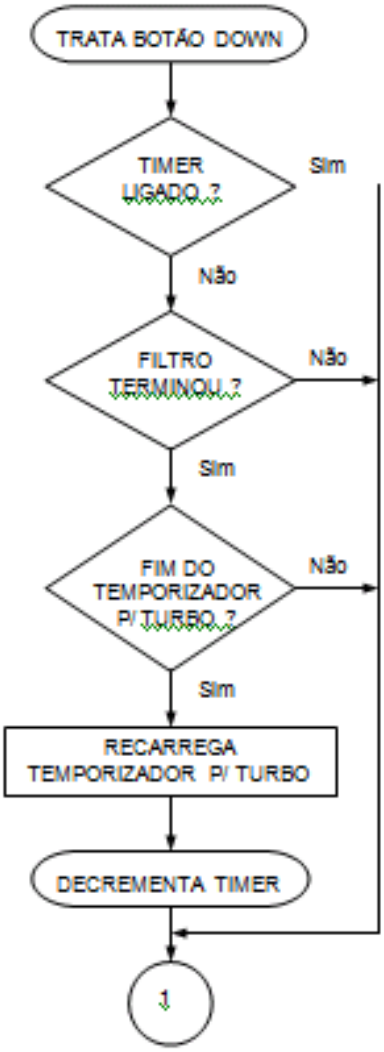


11.4. Fluxograma




INTERRUPÇÕES
SALVA CONTEXTO
INT TMR0 ?
Não
Sim
RECARREGA CONTADORES DO TMR1
SALVA O STATUS DOS BOTÕES
Sim
LINHA 4 HABILITADA ?
Não
PASSOU 1seg. ?
Não
Sim
INCREMENTA PONTEIRO (INDICE_VARRE_DISPLAY)
RECARREGA TEMPORIZADOR DE 1seg.
ACERTA PONTEIRO DO END. INDIRETO (FSR)
DECREMENTA TIMER
LIMPA TODOS OS SEGMENTOS (BLANK)
FIM DO TIMER ?
Não
DESLIGA TODOS OS DISPLAYS
Sim
CONSULTA TABELA E ACIONA DISPLAY CORRETO
LIMPA FLAG DE TIMER LIGADO (LED)
CONSULTA TABELA P/ DISPLAY DE 7 SEGMENTOS
PARA CONTADOR DO TIMER 1
ATUALIZA PORT LIGADO AOS SEGMENTOS
LIMPA FLAG DA INTERRUPÇÃO DE TMR1
LIMPA FLAG DA INTERRUPÇÃO DE TMR0
RECUPERA CONTEXTO
SAI DA INTERRUPÇÃO


TRATA BOTÃO UP
TIMER LIGADO ?
Sim
Não
FILTRO TERMINOU ?
Não
Sim
FIM DO TEMPORIZADOR P/ TURBO ?
Não
Sim
RECARREGA TEMPORIZADOR P/ TURBO
INCREMENTA TIMER
1


TRATA BOTÃO DOWN
TIMER LIGADO ?
Sim
Não
FILTRO TERMINOU ?
Não
Sim
FIM DO TEMPORIZADOR P/ TURBO ?
Não
Sim
RECARREGA TEMPORIZADOR P/ TURBO
DECREMENTA TIMER
1

11.5. Código

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Exemplo de Código para utilização com a McMaster            *
 *                                Exemplo 9                                 *
 *                                                                          *
 *                                 MOSAICO                                  *
 *                                                                          *
 *  TEL: (0XX11) 4992-8775                SITE: www.mosaico.com.br          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                           *
 *   DATA : 18/01/2010                                                      *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              Descrição geral                             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */


//ESTE EXEMPLO FOI PREPARADO PARA DEMOSNTRAR O FUNCIONAMENTO DO TIMER DE
//16 BITS DO PIC (TMR1) E DA VARREDURA DE DISPLAYS MAIS BARRA DE LEDS.
//CONSISTE NU       DOR DE SEGUNDOS. DOIS BOTÕES FC       LIZADOS PARA
//PROGRAMAR O TEMPO DA CONT       M OUTRO BOTÃO FOI UTILIZADO PA       ARAR
//O CONTADOR. O TEMPORIZADOR CONSEGUE CONTAR ATÉ 9999 SEGUNDOS, DE FORMA QUE
//OS 4 DISPLAYS DE 7 SEGMENTOS FORAM NECESSÁRIOS. A CONTAGEM É REGRESSIVA.
//UMA BARRA DE LEDS INDICA QUE O TEMPORIZADOR ESTÁ OPERANDO. QUANDO O SISTEMA
//CHEGA A 0000 A BARRA DE LEDS É DESLIGADA AUTOMATICAMENTE.


// BOTÕES DA LINHA 4 -> COLUNA 1 (INCREMENTA)
//                   -> COLUNA 2 (DECREMENTA)
//                   -> COLUNA 3 (LIGA CONTAGEM)
```

```
//                    -> COLUNA 4 (DESLIGA CONTAGEM)


/*     CONFIGURAÇÂO DOS JUMPERS DA PLACA:


               CAP USB(RC3)               -> ON

               DATA I2C                   -> OFF

               CLOCK I2C                  -> OFF

               COLUNA 2(RB1)              -> ON

               COLUNA 1(RB0)              -> ON

               LINHA 1 / DISPLAY MILHAR(RB4)    -> ON

               LEDS ESPECIAS (RC0,RC1,RC2)      -> ON


*/



/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              DEFINIÇÃO PIC                            *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */


#include <p18F4550.h>           // Register definitions


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                INCLUDES DAS FUNÇÕES DE PERIFÉRICOS DO PIC             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */


#include <pwm.h>                //PWM library functions
#include <adc.h>                //ADC library functions
#include <timers.h>             //Timer library functions
#include <delays.h>             //Delay library functions
#include <i2c.h>                //I2C library functions
#include <stdlib.h>             //Library functions
```

```
#include <usart.h>              //USART library functions


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Configurações para gravação                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */


#pragma config FOSC = HS
#pragma config CPUDIV = OSC1_PLL2
#pragma config WDT = ON
#pragma config WDTPS = 128
#pragma config LVP = OFF
#pragma config PWRT = ON
#pragma config BOR = OFF
#pragma config BORV = 0
#pragma config PBADEN = OFF
#pragma config DEBUG = OFF
#pragma config PLLDIV = 1
#pragma config USBDIV = 1
#pragma config FCMEN = OFF
#pragma config IESO = OFF
#pragma config VREGEN = OFF
#pragma config MCLRE = ON
#pragma config LPT1OSC = OFF
#pragma config CCP2MX = ON
#pragma config STVREN = OFF
#pragma config ICPRT = OFF


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Definição e inicialização das variáveis Globais           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Constantes internas                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de constantes facilita a programação e a manutenção.
#define t_filtro          240
#define turbo_tecla       2
#define delta_timer1     (65536 - 62500)


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Definição e inicialização das variáveis globais          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.
unsigned char display = 0;              // atualiza os displays
unsigned char contador = 2;             // contador de interrupções de timer1
unsigned char filtro = t_filtro;        // inicia filtro dos botões
unsigned char filtro1 = t_filtro;       // inicia filtro dos botões
unsigned char filtro2= t_filtro;        // inicia filtro dos botões
unsigned char filtro3 = t_filtro;       // inicia filtro dos botões
unsigned char filtro4 = t_filtro;       // inicia filtro dos botões
unsigned char turbo = 1;                // inicia turbo das teclas
unsigned char unidade = 9;              // variável unidade do timer de 1 seg
unsigned char dezena = 9;               // variável dezena do timer de 1 seg
unsigned char centena = 9;              // variável centena do timer de 1 seg
unsigned char milhar = 9;               // variável milhar do timer de 1 seg
unsigned char botoes = 0;


// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
// *                    TABELA DE CONVERSÃO BINÁRIO -> DISPLAY              *
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
// ESTA ROTINA IRÁ RETORNAR EM W, O SIMBOLO CORRETO QUE DEVE SER
```

```
// MOSTRADO NO DISPLAY PARA CADA VALOR DE INTENSIDADE. O RETORNO JÁ ESTÁ
// FORMATADO PARA AS CONDIÇÕES DE LIGAÇÃO DO DISPLAY AO PORTD.
//          a
//     **********
//     *        *
//   f *        * b
//     *    g   *
//     **********
//     *        *
//   e *        * c
//     *    d   *
//     **********  *.

const rom unsigned char converte[] =
{//   .GFEDCBA      POSIÇÃO CORRETA DOS SEGMENTOS
    0b00111111,     // 00 - RETORNA SÍMBOLO CORRETO 0
    0b00000110,     // 01 - RETORNA SÍMBOLO CORRETO 1
    0b01011011,     // 02 - RETORNA SÍMBOLO CORRETO 2
    0b01001111,     // 03 - RETORNA SÍMBOLO CORRETO 3
    0b01100110,     // 04 - RETORNA SÍMBOLO CORRETO 4
    0b01101101,     // 05 - RETORNA SÍMBOLO CORRETO 5
    0b01111101,     // 06 - RETORNA SÍMBOLO CORRETO 6
    0b00000111,     // 07 - RETORNA SÍMBOLO CORRETO 7
    0b01111111,     // 08 - RETORNA SÍMBOLO CORRETO 8
    0b01101111,     // 09 - RETORNA SÍMBOLO CORRETO 9
    0b0000000};    // Blank

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Declaração dos flags de software                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de flags ajuda na programação e economiza memória RAM.
```

```
struct
{
    unsigned BIT0:1;
    unsigned BIT1:1;
    unsigned BIT2:1;
    unsigned BIT3:1;
    unsigned BIT4:1;
    unsigned BIT5:1;
    unsigned BIT6:1;
    unsigned BIT7:1;
}FLAGSbits;             //ARMAZENA OS FLAGS DE CONTROLE

#define    estado_timer    FLAGSbits.BIT0        //STATUS DO TIMER 1

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Definição e inicialização dos port's               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        PROTOTIPAGEM DE FUNÇÕES                      *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void TRATA_HIGH_INT(void);
void TRATA_INT_TIMER0(void);
void TRATA_INT_TIMER1(void);
void decrementa_timer(void);
void incrementa_timer(void);

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                                ENTRADAS                             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
```

```
// AS SAÍDAS DEVEM SER ASSOCIADAS A NOMES PARA FACILITAR A PROGRAMAÇÃO E
//FUTURAS ALTERAÇÕES DO HARDWARE.



#define BT_UP              PORTBbits.RB0//BT_UP      //1 -> PRESSIONADO
                                                     //0 -> LIBERADO

#define BT_DOWN            PORTBbits.RB1//BT_DOWN    //1 -> PRESSIONADO
                                                     //0 -> LIBERADO

#define BT_START           PORTBbits.RB2//BT_START   //1 -> PRESSIONADO
                                                     //0 -> LIBERADO

#define BT_STOP            PORTBbits.RB3//BT_STOP     //1 -> PRESSIONADO
                                                     //0 -> LIBERADO


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                                SAÍDAS                                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// AS SAÍDAS DEVEM SER ASSOCIADAS A NOMES PARA FACILITAR A PROGRAMAÇÃO E
//FUTURAS ALTERAÇÕES DO HARDWARE.


#define      ESTADO_TIMER LATCbits.LATC0      // LED INDICATIVO DE TIMER LIGADO


#define      disp0  LATBbits.LATB4      // seleção do display unidade    (0)
#define      disp1  LATBbits.LATB5      // seleção do display dezena     (1)
#define      disp2  LATBbits.LATB6      // seleção do display centena    (2)
#define      disp3  LATBbits.LATB7      // seleção do display milhar     (3)


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Função de decremento do Timer                        *
```

```
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void decrementa_timer(void)
{
       unidade --;                      //DECREMENTA UNIDADE
       if (unidade == 0xFF)       //UNIDADE = 255?
       {
              unidade = 9;        //UNIDADE = 9
              dezena --;                 //DECREMENTA DEZENA
       }
       if (dezena == 0xFF)        //DEZENA = 255?
       {
              dezena = 9;                //DEZENA = 9
              centena --;                //DECREMENTA CENTENA
       }
       if (centena == 0xFF)       //CENTENA = 255?
       {
              centena = 9;        //CENTENA = 9
              milhar --;                 //DECREMENTA MILHAR
       }
       if (milhar == 0xFF)        //MILHAR = 255?
       {
              milhar = 9;                //MILHAR = 9
       }
}
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Função de incremento do Timer                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void incrementa_timer(void)
{
    unidade ++;
    if (unidade == 10)
```

```
    {
        unidade = 0;
        dezena ++;
    }
    if (dezena == 10)
    {
        dezena = 0;
        centena ++;
    }
    if (centena == 10)
    {
        centena = 0;
        milhar ++;
    }
    if (milhar == 10)
    {
        milhar = 0;
    }
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Função de tratamento de interrupção de Timer1          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Esta interrupcao ocorrera a cada 0,5 segundos.

void TRATA_INT_TIMER1(void)
{
    PIR1bits.TMR1IF = 0;            // limpa flag de interrupção
       WriteTimer1(delta_timer1);       // carraga timer1
    contador --;                    // decrementa contador de interrupções
    if (contador == 0)
```

```
    {
        WriteTimer1(delta_timer1);  // carraga timer1
        contador = 2;               // carrega contador de int
        decrementa_timer();
        if ((unidade == 0)&&(dezena == 0)&&(centena == 0)&&(milhar == 0))   // timer está
zerado?
        {
            estado_timer = 0;       // sim, apaga o led e
            CloseTimer1();
        }
    }
}
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Rotina de Tratamento de interrupção de TMR0             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Esta interrupção ocorrerá a cada 256us.

void TRATA_INT_TIMER0(void)
{
    INTCONbits.TMR0IF = 0;              //LIMPA FLAG DE INTERRUPÇÃO
    switch(display)                     // início da varredura dos display´s
    {
        case 0:
            display++;                  // incrementa a variável de varredura
            C_LEDS = 0;                 // liga o display 3
            PORTD = converte[milhar];  // atualiza o portd
            disp0 = 1;                  // liga o display 0
            break;                      // sai

        case 1:
            display++;                  // incrementa a variável de varredura
            disp0 = 0;                  // desliga o display 0
```

```
            PORTD = converte[centena];// atualiza o portd
            disp1 = 1;                  // liga o display 1
            break;                      // sai

        case 2:
            display++;                  // incrementa a variável de varredura
            disp1 = 0;                  // desliga o display 1
            PORTD = converte[dezena];  // atualiza o portd
            disp2 = 1;                  // liga o display 2
            break;                      // sai

        case 3:
            display = 0;        // inicia em 0 a variável de varredura
            disp2 = 0;                  // desliga o display 2
            PORTD = converte[unidade];// atualiza o portd
            disp3 = 1;                  // liga o display 3
            break;                      // sai
    }

}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função Principal                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void main ()
{
    PORTA = 0x00;               //Limpa PORTA
    PORTB = 0x00;               //Limpa PORTB
    PORTC = 0x00;               //Limpa PORTC
    PORTD = 0x00;               //Limpa PORTD
    PORTE = 0x00;               //Limpa PORTE
```

```
    LATA = 0x00;                //Limpa PORTA
    LATB = 0x00;                //Limpa PORTB
    LATC = 0x00;                //Limpa PORTC
    LATD = 0x00;                //Limpa PORTD
    LATE = 0x00;                //Limpa PORTE

    TRISA = 0b11111111;                 //CONFIG DIREÇÃO DOS PINOS PORTA
    TRISB = 0b00001111;         //CONFIG DIREÇÃO DOS PINOS PORTB
    TRISC = 0b11111000;                     //CONFIG DIREÇÃO DOS PINOS PORTC
    TRISD = 0b00000000;         //CONFIG DIREÇÃO DOS PINOS PORTD
     TRISE = 0b00000111;                //CONFIG DIREÇÃO DOS PINOS PORTE

    ADCON1 = 0b00001111;        //DESLIGA CONVERSORES A/D

       OpenTimer0(TIMER_INT_ON & T0_8BIT & T0_SOURCE_INT & T0_PS_1_2);
//     //CONFIGURAÇÃO DO TIMER0
       OpenTimer1(T1_OSC1EN_OFF & TIMER_INT_OFF & T1_SOURCE_INT & T1_8BIT_RW & T1_PS_1_8);
//     //CONFIGURAÇÃO DO TIMER1
//
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Inicialização do Sistema                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

       while(RCONbits.NOT_TO);    //AGUARDA ESTOURO DO WDT

       ESTADO_TIMER = 0;          //LIMPA ESTADO_TIMER

       INTCONbits.PEIE = 1;       //LIGA CHAVE DOS PERIFÉRICOS
       INTCONbits.GIE = 1;        //LIGA A CHAVE GERAL
       CloseTimer1();
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Rotina Principal                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Loop principal                            *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
loop:
    while(1)
    {
        ClrWdt();
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                         Tratamento do Botão 1                      *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
             if(BT_UP)                  // testa botão 1
        {                               // botão 1 está pressionado ?
                    if(ESTADO_TIMER == 0)           // o timer está parado?
        {
                        filtro1 --;         // decrementa o filtro
                        if(filtro1 == 0)    // fim do filtro do botão?
            {
                turbo --;               // decrementa o turbo da tecla
                if (turbo == 0)         // sim, fim do turbo do botão ?
                {
                    turbo = turbo_tecla;// carrega o turbo
                    incrementa_timer(); // incrementa o timer
                }
            }
        }
```

```
                continue;
            }

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      Tratamento do Botão 2                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

            if(BT_DOWN)                 // testa botão 2
            {                           // botão 2 está pressionado ?
                if(ESTADO_TIMER == 0)           // o timer está parado?
        {
                    filtro2 --;         // decrementa o filtro
                    if(filtro2 == 0)    // fim do filtro do botão?
            {
                turbo --;               // decrementa o turbo da tecla
                if (turbo == 0)         // sim, fim do turbo do botão ?
                {
                    turbo = turbo_tecla;// carrega o turbo
                    decrementa_timer(); // decrementa o timer
                }
            }
        }
                    continue;
            }

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      Tratamento do Botão 3                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

            if(BT_START)                // testa botão 3
            {                           // botão 3 está pressionado ?
```

```
                if (ESTADO_TIMER == 0)
                {
                        filtro3 --;          // decrementa o filtro
                        if(filtro3 == 0)    // fim do filtro do botão?
            {
                if   ((unidade!=0)||(dezena!=0)||(centena!=0)||(milhar!=0))     //   timer   está
zerado?
                {
                     ESTADO_TIMER = 1; // não, indica que o timer está contando
                    contador = 2;        // carrega contador auxiliar
                    WriteTimer1(delta_timer1);    // carraga timer1
                     PIR1bits.TMR1IF = 0;            // limpa flag de int tmr1
                     OpenTimer1(T1_OSC1EN_OFF  &  TIMER_INT_ON  &  T1_SOURCE_INT  &  T1_8BIT_RW  &
T1_PS_1_8);  // liga o tratamento de interrupção de timer1

                }
            }
            continue;
                }
            }

            if(BT_STOP)                // testa botão 4
            {
                ESTADO_TIMER = 0;       // indica timer parado
                CloseTimer1();   // desliga o tratamento de interrupção de timer1
            }

        }
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
// *             ROTINA DE TRATAMENTO DE INT DE ALTA PRIORIDADE              *
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
#pragma code VETOR_HIGH_PRIORITY = 0x0008     //VETOR DE ALTA PRIORIDADE

void HIGH_int (void)

{

  _asm goto TRATA_HIGH_INT _endasm

}

#pragma code


#pragma interrupt TRATA_HIGH_INT

void TRATA_HIGH_INT(void)

{

    if(INTCONbits.TMR0IF)     TRATA_INT_TIMER0();

    if(PIR1bits.TMR1IF)       TRATA_INT_TIMER1();

}
```

11.6. Dicas e Comentários

Observar que nesta experiência, ao entrar no tratamento das interrupções, a operação de salvar contexto é maior que nas experiências anteriores. Isto por que agora salva-se também os valores de FSR e PCLATH pois os mesmos podem ser alterados dentro da interrupção.

11.7. Exercícios Propostos

1. Implemente o quarto botão, para resetar o temporizador (voltar a zero).
2. Implemente uma segunda velocidade para os botões de incremento e decremento, de forma que facilite o ajuste de valores maiores.
3. Em vez de fazer um timer somente de segundos, utilize os dois dígitos da esquerda para mostrar o tempo em minutos e os da direita para mostrar o tempo em segundos. O ponto do display da centena pode ser usado para marcar a separação. Não se esqueça que agora os displays da direita devem contar somente de 0 a 59 e não mais de 0 a 99.

12. Experiência 10 – Display de cristal líquido LCD

12.1. Objetivo

O objetivo desta experiência é o aprendizado da utilização de display de cristal líquido.

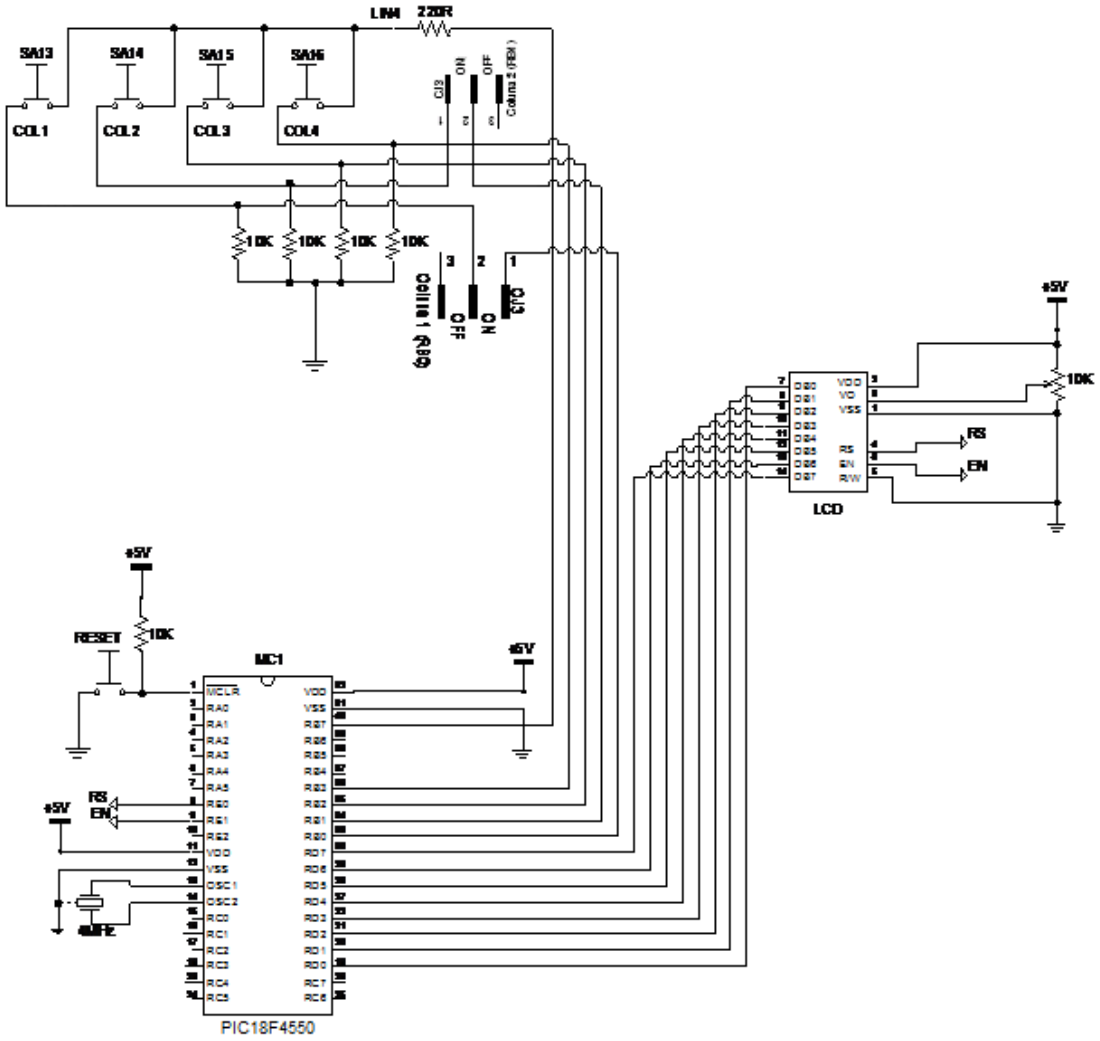
12.2. Descrição

Esta experiência foi elaborada para explicar o funcionamento do display LCD e o exemplo de software proposto é bastante reduzido. Simplesmente utilizou-se o LCD para informar ao usuário qual tecla está pressionada.

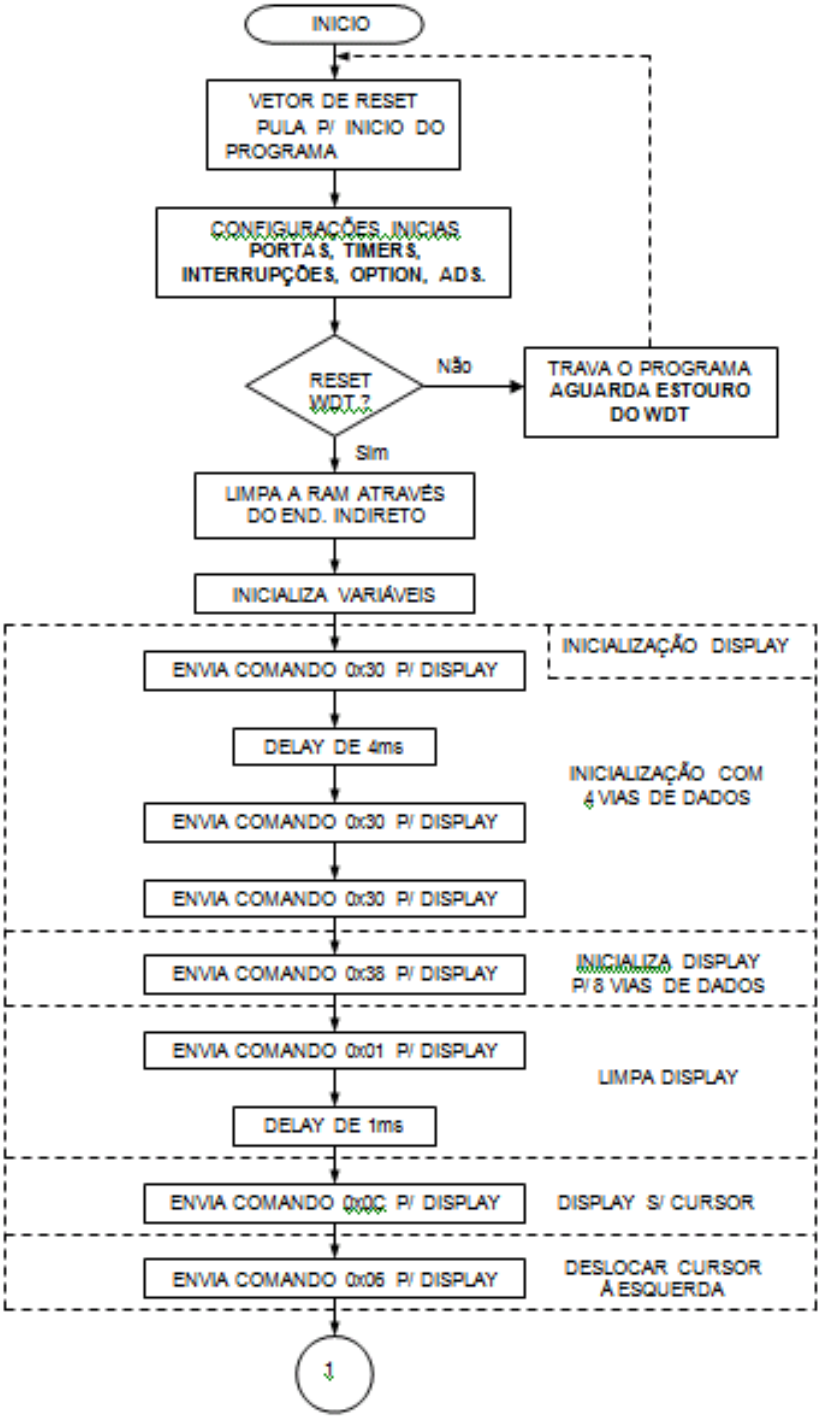
Para isso elaborou-se uma rotina chamada ESCREVE que envia a informação passada pelo Work ao display. Esta rotina pode ser utilizada tanto para enviar comandos quanto dados. Foi criada também, a rotina de inicialização do LCD. Esta rotina configura o sistema para comunicação com 8 vias, 2 linhas, sem cursor visível e com movimento automático do cursor para a direita. Além disso, ela já limpa a tela e posiciona o cursor na primeira linha, primeiro caractere da esquerda.

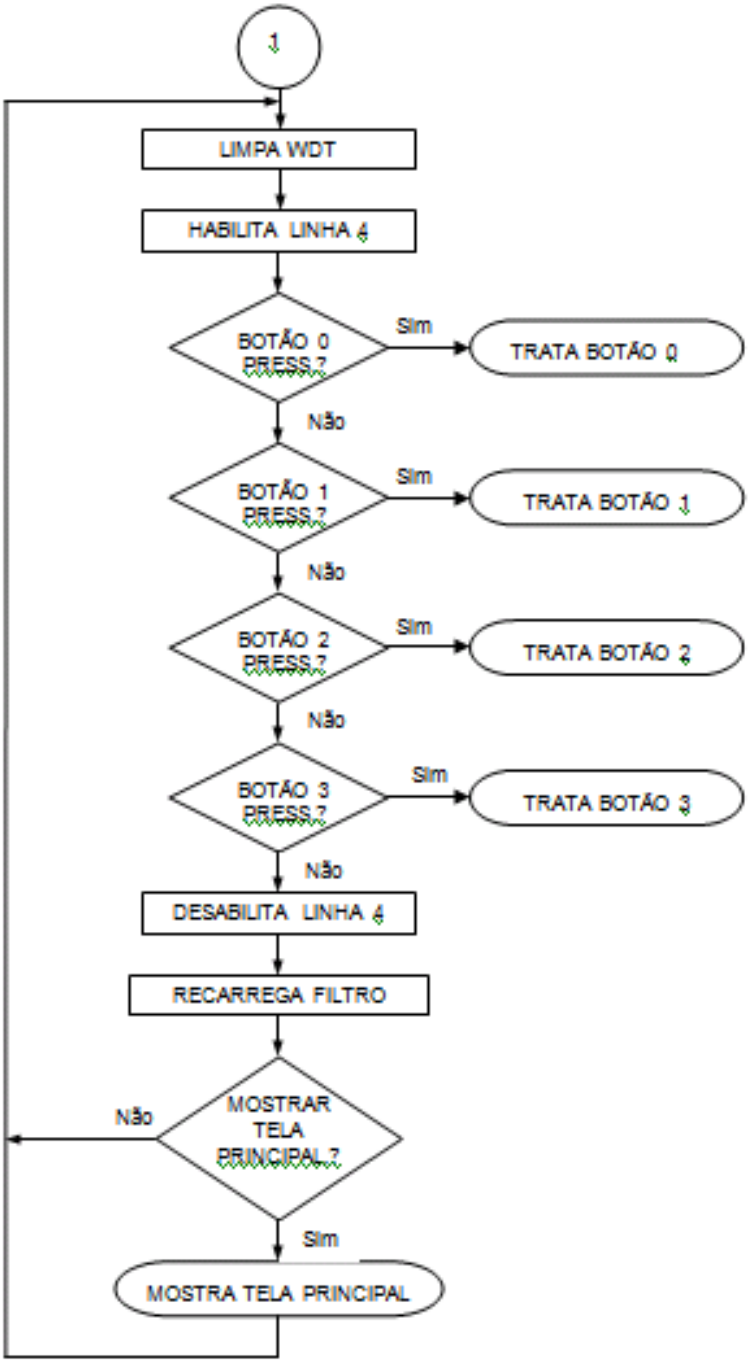
Para cada botão pressionado, posicionou-se o cursor em um local diferente da tela e escreveu-se o número do botão em questão. Após a liberação, uma tela padrão é visualizada.

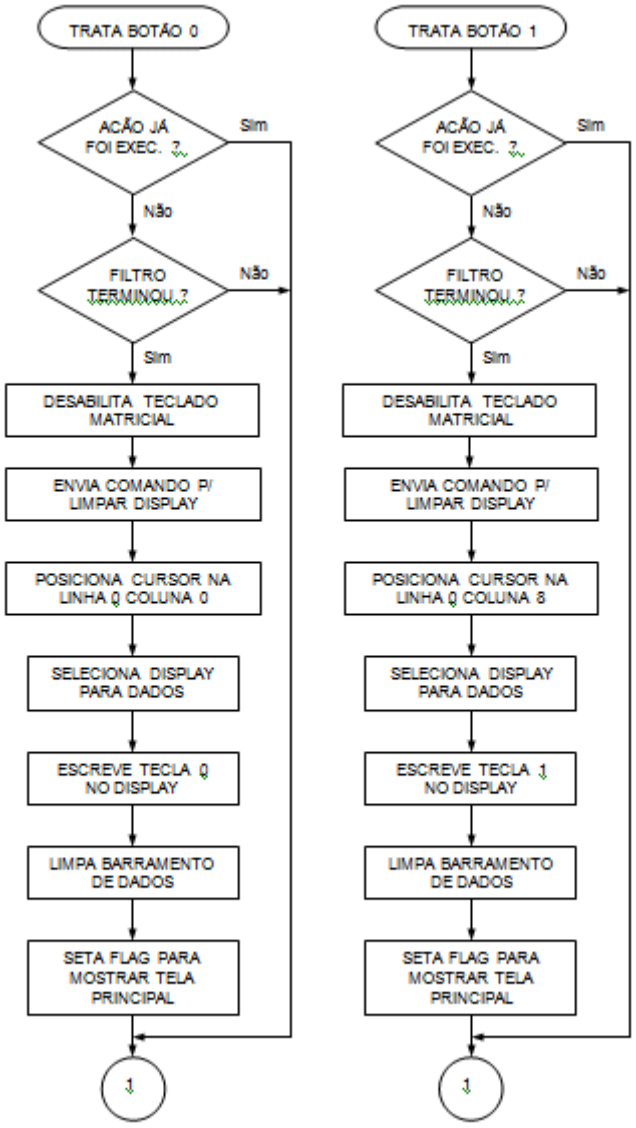
12.3. Esquema Elétrico

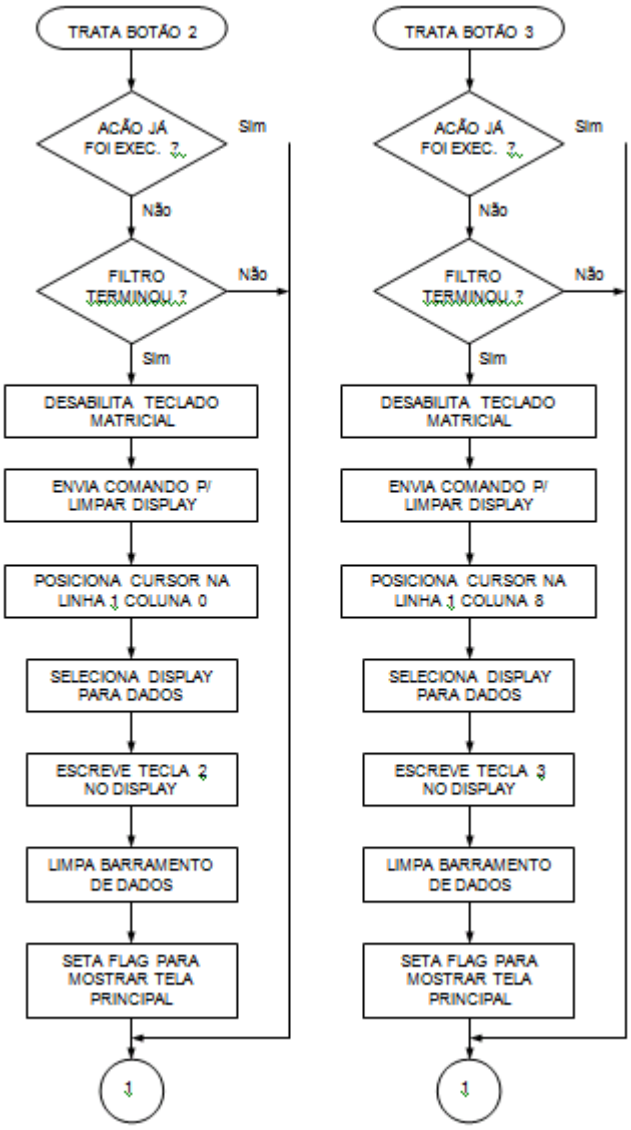


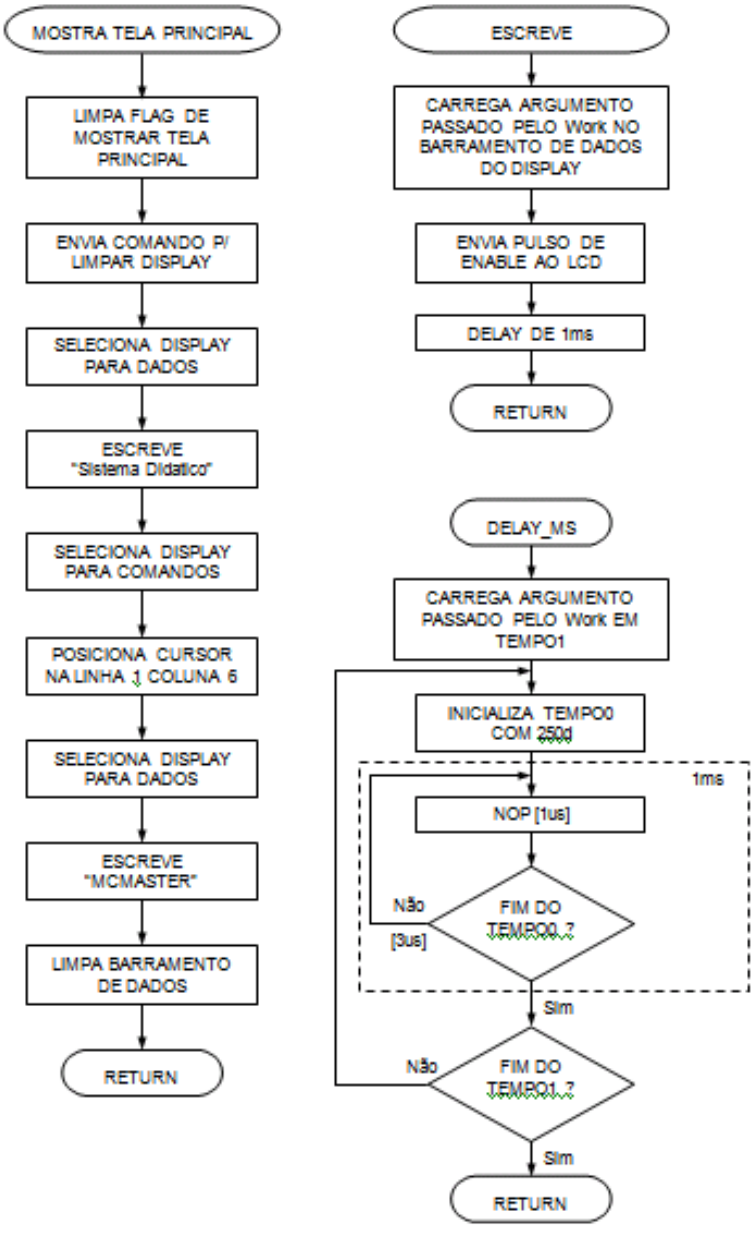
12.4. Fluxograma


INICIO
VETOR DE RESET PULA P/ INICIO DO PROGRAMA
CONFIGURAÇÕES INICIAS PORTAS, TIMERS, INTERRUPÇÕES, OPTION, ADS.
RESET WDT ?
Não
TRAVA O PROGRAMA AGUARDA ESTOURO DO WDT
Sim
LIMPA A RAM ATRAVÉS DO END. INDIRETO
INICIALIZA VARIÁVEIS
INICIALIZAÇÃO DISPLAY
ENVIA COMANDO 0x30 P/ DISPLAY
DELAY DE 4ms
ENVIA COMANDO 0x30 P/ DISPLAY
ENVIA COMANDO 0x30 P/ DISPLAY
INICIALIZAÇÃO COM 4 VIAS DE DADOS
ENVIA COMANDO 0x38 P/ DISPLAY
INICIALIZA DISPLAY P/ 8 VIAS DE DADOS
ENVIA COMANDO 0x01 P/ DISPLAY
DELAY DE 1ms
LIMPA DISPLAY
ENVIA COMANDO 0x0C P/ DISPLAY
DISPLAY S/ CURSOR
ENVIA COMANDO 0x06 P/ DISPLAY
DESLOCAR CURSOR À ESQUERDA
1


1
LIMPA WDT
HABILITA LINHA 4
BOTÃO 0 PRESS.?
Sim
TRATA BOTÃO 0
Não
BOTÃO 1 PRESS.?
Sim
TRATA BOTÃO 1
Não
BOTÃO 2 PRESS.?
Sim
TRATA BOTÃO 2
Não
BOTÃO 3 PRESS.?
Sim
TRATA BOTÃO 3
Não
DESABILITA LINHA 4
RECARREGA FILTRO
MOSTRAR TELA PRINCIPAL ?
Não
Sim
MOSTRA TELA PRINCIPAL


TRATA BOTÃO 0
AÇÃO JÁ FOI EXEC. ?
Sim
Não
FILTRO TERMINOU ?
Não
Sim
DESABILITA TECLADO MATRICIAL
ENVIA COMANDO P/ LIMPAR DISPLAY
POSICIONA CURSOR NA LINHA 0 COLUNA 0
SELECIONA DISPLAY PARA DADOS
ESCREVE TECLA 0 NO DISPLAY
LIMPA BARRAMENTO DE DADOS
SETA FLAG PARA MOSTRAR TELA PRINCIPAL
1
TRATA BOTÃO 1
AÇÃO JÁ FOI EXEC. ?
Sim
Não
FILTRO TERMINOU ?
Não
Sim
DESABILITA TECLADO MATRICIAL
ENVIA COMANDO P/ LIMPAR DISPLAY
POSICIONA CURSOR NA LINHA 0 COLUNA 8
SELECIONA DISPLAY PARA DADOS
ESCREVE TECLA 1 NO DISPLAY
LIMPA BARRAMENTO DE DADOS
SETA FLAG PARA MOSTRAR TELA PRINCIPAL
1


TRATA BOTÃO 2
AÇÃO JÁ FOI EXEC. ?
Sim
Não
FILTRO TERMINOU ?
Não
Sim
DESABILITA TECLADO MATRICIAL
ENVIA COMANDO P/ LIMPAR DISPLAY
POSICIONA CURSOR NA LINHA 1 COLUNA 0
SELECIONA DISPLAY PARA DADOS
ESCREVE TECLA 2 NO DISPLAY
LIMPA BARRAMENTO DE DADOS
SETA FLAG PARA MOSTRAR TELA PRINCIPAL
1
TRATA BOTÃO 3
AÇÃO JÁ FOI EXEC. ?
Sim
Não
FILTRO TERMINOU ?
Não
Sim
DESABILITA TECLADO MATRICIAL
ENVIA COMANDO P/ LIMPAR DISPLAY
POSICIONA CURSOR NA LINHA 1 COLUNA 8
SELECIONA DISPLAY PARA DADOS
ESCREVE TECLA 3 NO DISPLAY
LIMPA BARRAMENTO DE DADOS
SETA FLAG PARA MOSTRAR TELA PRINCIPAL
1


MOSTRA TELA PRINCIPAL
LIMPA FLAG DE MOSTRAR TELA PRINCIPAL
ENVIA COMANDO P/ LIMPAR DISPLAY
SELECIONA DISPLAY PARA DADOS
ESCREVE "Sistema Didatico"
SELECIONA DISPLAY PARA COMANDOS
POSICIONA CURSOR NA LINHA 1 COLUNA 6
SELECIONA DISPLAY PARA DADOS
ESCREVE "MCMASTER"
LIMPA BARRAMENTO DE DADOS
RETURN
ESCREVE
CARREGA ARGUMENTO PASSADO PELO Work NO BARRAMENTO DE DADOS DO DISPLAY
ENVIA PULSO DE ENABLE AO LCD
DELAY DE 1ms
RETURN
DELAY_MS
CARREGA ARGUMENTO PASSADO PELO Work EM TEMPO1
INICIALIZA TEMPO0 COM 250d
1ms
NOP [1us]
Não
[3us]
FIM DO TEMPO0 ?
Sim
Não
FIM DO TEMPO1 ?
Sim
RETURN

12.5. Código

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Exemplo de Código para utilização com a McMaster             *
 *                                 Exemplo 10                                *
 *                                                                           *
 *                                  MOSAICO                                  *
 *                                                                           *
 *  TEL: (0XX11) 4992-8775                 SITE: www.mosaico.com.br          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                            *
 *   DATA : 18/01/2010                                                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              Descrição geral                              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

//ESTE EXEMPLO FOI ELABORADO PARA EXPLICAR O FUNCIONAMENTO DO MÓDULO DE LCD.
//FOI CRIADA UMA ROTINA PARA ESCREVER COMANDOS OU CARACTERES NO LCD. EXISTE
//TAMBÉM UMA ROTINA DE INICIALIZAÇÃO NECESSÁRIA PARA A CORRETA CONFIGURAÇÃO
//DO LCD. OS BOTÕES CONTINUAM SENDO MONITORADOS. UMA MENSAGEM É ESCRITA
//NO LCD PARA CADA UM DOS BOTÕES, QUANDO O MESMO É PRESSIONADO.
/*
 CONFIGURAÇÂO DOS JUMPERS DA PLACA:

        CAP USB(RC3)                -> ON
        DATA I2C                    -> OFF
        CLOCK I2C                   -> OFF
        COLUNA 2(RB1)               -> ON
        COLUNA 1(RB0)               -> ON
        LINHA 1 / DISPLAY MILHAR(RB4)    -> ON
        LEDS ESPECIAS (RC0,RC1,RC2)      -> ON

*/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                             DEFINIÇÃO PIC                                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <p18F4550.h>           // Register definitions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               INCLUDES DAS FUNÇÕES DE PERIFÉRICOS DO PIC                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <pwm.h>                //PWM library functions
#include <adc.h>                //ADC library functions
#include <timers.h>             //Timer library functions
#include <delays.h>             //Delay library functions
#include <i2c.h>                //I2C library functions
#include <stdlib.h>             //Library functions
#include <usart.h>              //USART library functions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Configurações para gravação                         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#pragma config FOSC = HS
#pragma config CPUDIV = OSC1_PLL2
#pragma config WDT = OFF
```

```
#pragma config WDTPS = 128
#pragma config LVP = OFF
#pragma config PWRT = ON
#pragma config BOR = OFF
#pragma config BORV = 0
#pragma config PBADEN       = OFF
#pragma config DEBUG        = OFF
#pragma config PLLDIV       = 1
#pragma config USBDIV       = 1
#pragma config FCMEN        = OFF
#pragma config IESO  = OFF
#pragma config VREGEN       = OFF
#pragma config MCLRE        = ON
#pragma config LPT1OSC      = OFF
#pragma config CCP2MX = ON
#pragma config STVREN       = OFF
#pragma config ICPRT        = OFF

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Definição e inicialização das variáveis Globais          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.
unsigned char    FILTRO11;        //FILTRAGEM  1 PARA O BOTÃO 1
unsigned char    FILTRO12;        //FILTRAGEM  2 PARA O BOTÃO 1
unsigned char    FILTRO21;        //FILTRAGEM  1 PARA O BOTÃO 2
unsigned char    FILTRO22;        //FILTRAGEM  2 PARA O BOTÃO 2
unsigned char    FILTRO31;        //FILTRAGEM  1 PARA O BOTÃO 3
unsigned char    FILTRO32;        //FILTRAGEM  2 PARA O BOTÃO 3
unsigned char    FILTRO41;        //FILTRAGEM  1 PARA O BOTÃO 4
unsigned char    FILTRO42;        //FILTRAGEM  2 PARA O BOTÃO 4

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Constantes internas                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de constantes facilita a programação e a manutenção.
#define     T_FILTRO    25

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Declaração dos flags de software                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de flags ajuda na programação e economiza memória RAM.

struct {
    unsigned BIT0:1;
    unsigned BIT1:1;
    unsigned BIT2:1;
    unsigned BIT3:1;
    unsigned BIT4:1;
    unsigned BIT5:1;
    unsigned BIT6:1;
    unsigned BIT7:1;
}FLAGSbits;            //ARMAZENA OS FLAGS DE CONTROLE

#define    ST_BT1        FLAGSbits.BIT0        //STATUS DO BOTÃO 1
#define    ST_BT2        FLAGSbits.BIT1        //STATUS DO BOTÃO 2
#define    ST_BT3        FLAGSbits.BIT2        //STATUS DO BOTÃO 3
#define    ST_BT4        FLAGSbits.BIT3        //STATUS DO BOTÃO 4
#define    ST_NO_BOTS    FLAGSbits.BIT4        //STATUS DOS BOTÕES LIBERADOS


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                 Definição e inicialização dos port's                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       PROTOTIPAGEM DE FUNÇÕES                         *
```

```
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void comando_lcd(unsigned char caracter);
void escreve_lcd(unsigned char caracter);
void limpa_lcd(void);
void inicializa_lcd(void);
void tela_principal(void);

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              ENTRADAS                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// As entradas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define     BT1 PORTBbits.RB0   //PORTA DO BOTÃO
                                //1 -> PRESSIONADO
                                //0 -> LIBERADO

#define     BT2 PORTBbits.RB1   //PORTA DO BOTÃO
                                //1 -> PRESSIONADO
                                //0 -> LIBERADO

#define     BT3 PORTBbits.RB2   //PORTA DO BOTÃO
                                //1 -> PRESSIONADO
                                //0 -> LIBERADO

#define     BT4 PORTBbits.RB3   //PORTA DO BOTÃO
                                //1 -> PRESSIONADO
                                //0 -> LIBERADO

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                               SAÍDAS                                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//As saídas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define     LINHA_4     PORTBbits.RB7   //PINO PARA ATIVAR LINHA 4 DO TECLADO
                                        //MATRICIAL
                                        //1 -> LINHA ATIVADA
                                        //0 -> LINHA DESATIVADA

#define     rs      PORTEbits.RE0       // via do lcd que sinaliza recepção de dados ou
comando
#define     enable  PORTEbits.RE1       // enable do lcd

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Rotina que envia um COMANDO para o LCD                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void comando_lcd(unsigned char caracter)
{
    rs = 0;                     // seleciona o envio de um comando
    PORTD = caracter;           // carrega o PORTD com o caracter
    enable = 1 ;                // gera pulso no enable
    Delay10TCYx(1);             // espera 3 microsegundos
    enable = 0;                 // desce o pino de enable
    Delay10TCYx(4);             // espera mínimo 40 microsegundos
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                Rotina que envia um DADO a ser escrito no LCD            *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void escreve_lcd(unsigned char caracter)
{
    rs = 1;                     // seleciona o envio de um comando
```

```
    PORTD = caracter;           // carrega o PORTD com o caracter
    enable = 1;                 // gera pulso no enable
    Delay10TCYx(1);             // espera 10 microsegundos
    enable = 0;                 // desce o pino de enable
    Delay10TCYx(4);             // espera mínimo 40 microsegundos
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            Função para limpar o LCD                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void limpa_lcd(void)
{
    comando_lcd(0x01);          // limpa lcd
    Delay1KTCYx(2);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Inicialização do Display de LCD                      *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void inicializa_lcd(void)
{
    comando_lcd(0x30);          // envia comando para inicializar display
    Delay1KTCYx(4);             // espera 4 milisengundos

    comando_lcd(0x30);          // envia comando para inicializar display
    Delay10TCYx(10);            // espera 100 microsengundos

    comando_lcd(0x30);          // envia comando para inicializar display

    comando_lcd(0x38);          // liga o display, sem cursor e sem blink

    limpa_lcd();                // limpa lcd

    comando_lcd(0x0c);          // display sem cursor

    comando_lcd(0x06);          // desloca cursor para a direita
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Tela Principal                              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void tela_principal(void)
{
    limpa_lcd();                // limpa lcd
    comando_lcd(0x80);          // posiciona o cursor na linha 0, coluna 0
    escreve_lcd ('S');          // imprime mensagem no lcd
    escreve_lcd ('i');
    escreve_lcd ('s');
    escreve_lcd ('t');
    escreve_lcd ('e');
    escreve_lcd ('m');
    escreve_lcd ('a');
    escreve_lcd (' ');
    escreve_lcd ('D');
    escreve_lcd ('i');
    escreve_lcd ('d');
    escreve_lcd ('a');
    escreve_lcd ('t');
    escreve_lcd ('i');
    escreve_lcd ('c');
    escreve_lcd ('o');

    comando_lcd(0xC4);          // posiciona o cursor na linha 1, coluna 4
```

```
    escreve_lcd ('M');          // imprime mensagem no lcd
    escreve_lcd ('C');
    escreve_lcd ('M');
    escreve_lcd ('A');
    escreve_lcd ('S');
    escreve_lcd ('T');
    escreve_lcd ('E');
    escreve_lcd ('R');
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Tela Tecla 1                                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void tela_tecla_1(void)
{
    limpa_lcd();                // limpa lcd
    comando_lcd(0x80);          // posiciona o cursor na linha 0, coluna 0
    escreve_lcd ('T');          // imprime mensagem no lcd
    escreve_lcd ('E');
    escreve_lcd ('C');
    escreve_lcd ('L');
    escreve_lcd ('A');
    escreve_lcd (' ');
    escreve_lcd ('1');
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Tela Tecla 2                                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void tela_tecla_2(void)
{
    limpa_lcd();                // limpa lcd
    comando_lcd(0x89);          // posiciona o cursor na linha 0, coluna 0
    escreve_lcd ('T');          // imprime mensagem no lcd
    escreve_lcd ('E');
    escreve_lcd ('C');
    escreve_lcd ('L');
    escreve_lcd ('A');
    escreve_lcd (' ');
    escreve_lcd ('2');
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Tela Tecla 3                                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void tela_tecla_3(void)
{
    limpa_lcd();                // limpa lcd
    comando_lcd(0xC0);          // posiciona o cursor na linha 0, coluna 0
    escreve_lcd ('T');          // imprime mensagem no lcd
    escreve_lcd ('E');
    escreve_lcd ('C');
    escreve_lcd ('L');
    escreve_lcd ('A');
    escreve_lcd (' ');
    escreve_lcd ('3');
}


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Tela Tecla 4                                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
```

```
void tela_tecla_4(void)
{
    limpa_lcd();               // limpa lcd
    comando_lcd(0xC9);         // posiciona o cursor na linha 0, coluna 0
    escreve_lcd ('T');         // imprime mensagem no lcd
    escreve_lcd ('E');
    escreve_lcd ('C');
    escreve_lcd ('L');
    escreve_lcd ('A');
    escreve_lcd (' ');
    escreve_lcd ('4');
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função Principal                             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void main ()
{
    PORTA = 0x00;               //Limpa PORTA
    PORTB = 0x00;               //Limpa PORTB
    PORTC = 0x00;               //Limpa PORTC
    PORTD = 0x00;               //Limpa PORTD
    PORTE = 0x00;               //Limpa PORTE

    LATA = 0x00;                //Limpa PORTA
    LATB = 0x00;                //Limpa PORTB
    LATC = 0x00;                //Limpa PORTC
    LATD = 0x00;                //Limpa PORTD
    LATE = 0x00;                //Limpa PORTE

    TRISA = 0b00101111;         //CONFIG DIREÇÃO DOS PINOS PORTA
    TRISB = 0b00001111;         //CONFIG DIREÇÃO DOS PINOS PORTB
    TRISC = 0b10011001;         //CONFIG DIREÇÃO DOS PINOS PORTC
    TRISD = 0b00000000;         //CONFIG DIREÇÃO DOS PINOS PORTD
    TRISE = 0b00000000;         //CONFIG DIREÇÃO DOS PINOS PORTE

    ADCON1 = 0b00001111;        //DESLIGA CONVERSORES A/D

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Inicialização do Sistema                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

    RCON = 0x00;

    LINHA_4 = 1;                //ATIVA LINHA 4 DO TECLADO MATRICIAL

    inicializa_lcd();           // configura o lcd
    tela_principal();

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Rotina Principal                             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
    while(1)
    {
        ClrWdt();

        FILTRO11 = 0;           //BT1
        FILTRO12 = T_FILTRO;    //BT1

        FILTRO21 = 0;           //BT2
        FILTRO22 = T_FILTRO;    //BT2

        FILTRO31 = 0;           //BT3
        FILTRO32 = T_FILTRO;    //BT3
```

```
FILTRO41 = 0;             //BT4
FILTRO42 = T_FILTRO;      //BT4

if(!ST_NO_BOTS)
{
    tela_principal();
    ST_NO_BOTS = 1;
}

while(BT1)
{
    ClrWdt();
    if (--FILTRO11 == 0)
    {
        FILTRO12--;
        if (FILTRO12 == 0)
        {
            if (!ST_BT1)
            {
                tela_tecla_1();
                ST_BT1 = 1;
                ST_NO_BOTS = 0;
            }
        }
    }
}
ST_BT1 = 0; // BOTÃO LIBERADO, LIMPA O FLAG

while(BT2)
{
    ClrWdt();
    if (--FILTRO21 == 0)
    {
        FILTRO22--;
        if (FILTRO22 == 0)
        {
            if (!ST_BT2)
            {
                tela_tecla_2();
                ST_BT2 = 1;
                ST_NO_BOTS = 0;
            }
        }
    }
}
ST_BT2 = 0; // BOTÃO LIBERADO, LIMPA O FLAG


while(BT3)
{
    ClrWdt();
    if (--FILTRO31 == 0)
    {
        FILTRO32--;
        if (FILTRO32 == 0)
        {
            if (!ST_BT3)
            {
                tela_tecla_3();
                ST_BT3 = 1;
                ST_NO_BOTS = 0;
            }
        }
    }
}
ST_BT3 = 0; // BOTÃO LIBERADO, LIMPA O FLAG
```

```
            while(BT4)
            {
                ClrWdt();
                if (--FILTRO41 == 0)
                {
                    FILTRO42--;
                    if (FILTRO42 == 0)
                    {
                        if (!ST_BT4)
                        {
                            tela_tecla_4();
                            ST_BT4 = 1;
                            ST_NO_BOTS = 0;
                        }
                    }
                }
            }
            ST_BT4 = 0; // BOTÃO LIBERADO, LIMPA O FLAG

    }
}
```

12.6. Dicas e Comentários

Apesar da estrutura do sistema ficar muito simples com a implementação da rotina ESCREVE, nunca se deve esquecer de confirmar o estado da saída RS (define comando ou dado) antes de utilizá-la.

Notar que para enviar um caractere ao LCD deve-se utilizar o código ASCII do caractere.

Apesar do sistema McMaster possuir ligação com o módulo de LCD através de 8 vias de dados é possível utilizá-lo para testar e implementar a comunicação com 4 vias. Basta modificar a rotina de inicialização e a de escrita de um byte.

12.7. Exercícios Propostos

1. Altere a comunicação para 4 vias.
2. Mantenha a tela principal disponível somente quando o sistema é ligado. Após alguns segundos, mostre uma tela com o nome das quatro teclas e indique a tecla pressionada através de um caractere de seta (←) ou outro qualquer.

13. Experiência 11 – Conversor A/D

13.1. Objetivo

Nesta experiência será estudado o módulo de conversão A/D interno do PIC18F4550

13.2. Descrição

Este exemplo foi elaborado para explicar o funcionamento do módulo de conversão analógico digital interno do PIC18F4550. É convertido o valor analógico presente no pino RA0 do microcontrolador, sendo que este valor pode ser alterado através do potenciômetro presente na placa do sistema McMaster.

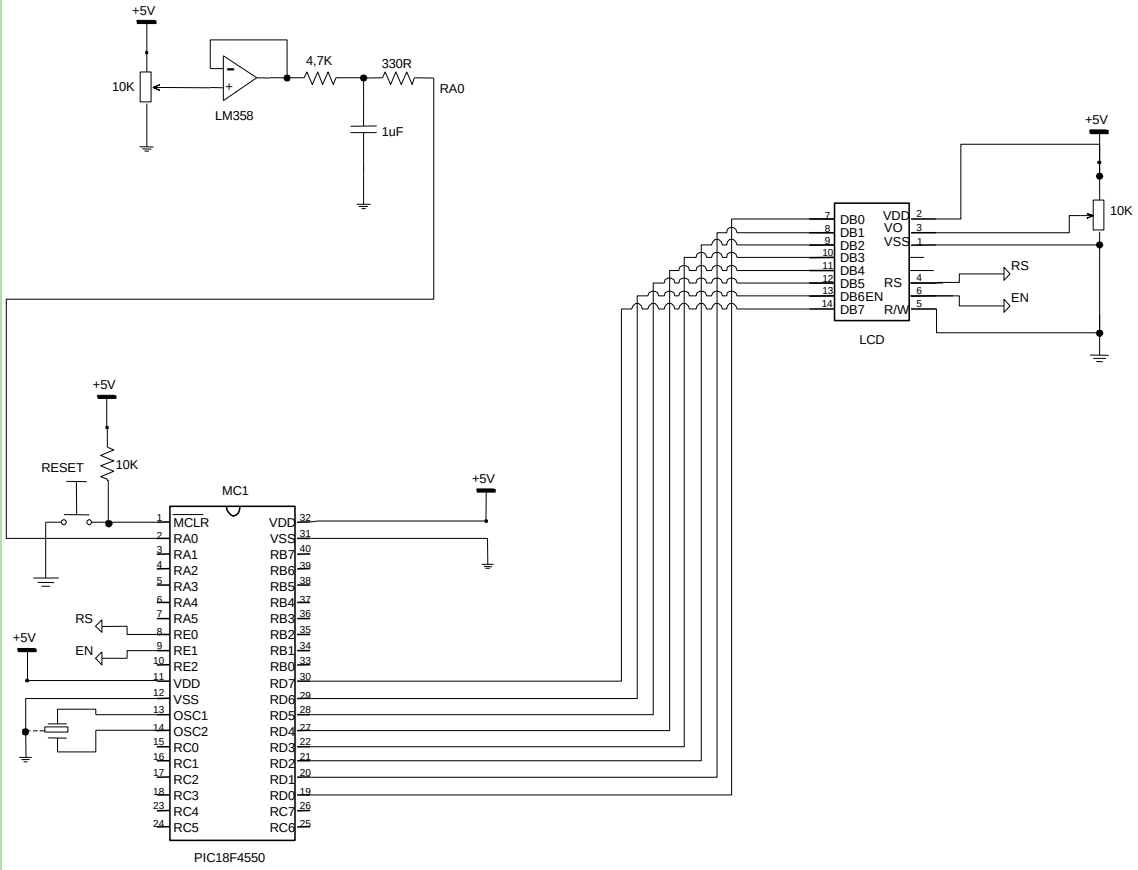
A conversão é feita diretamente no loop principal, sem a utilização de nenhuma interrupção, nem para checar o término da conversão nem para definir uma freqüência de amostragem. Desta forma, a conversão será feita uma após a outra, na freqüência definida pelo período do loop principal.

Uma vez terminada a conversão, descarta-se os 2 bits menos significativos e considera-se somente o resultado armazenado em **ADRESH**. Com isso já se está executando uma espécie de filtragem, evitando assim que o valor final fique oscilando.

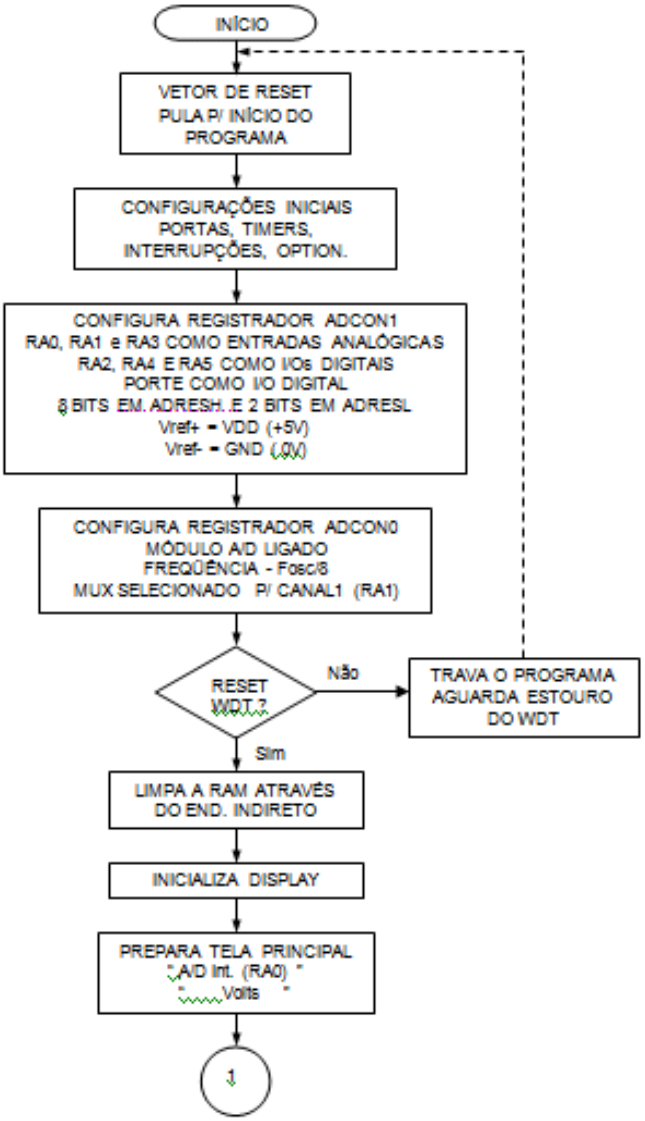
Aplica-se então uma regra de 3 para converter o valor do A/D para a unidade desejada: Volts. Considerando-se que quando o A/D resulta em 0 (zero) a entrada possui 0,0V, e quando o A/D resulta em 255 a entrada é equivalente a 5,0V, aplica-se a regra de 3 é mostra-se o valor da tensão, já em Volts, no LCD.

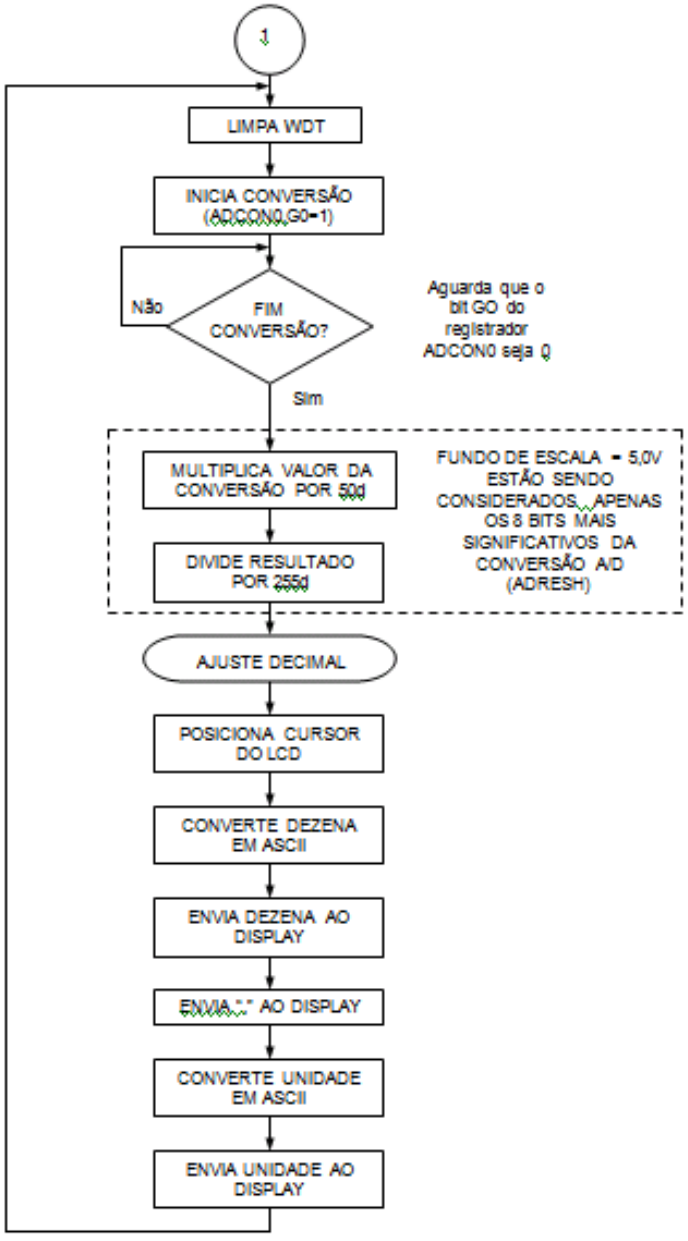
Para a execução da regra de 3 foram utilizadas rotinas de multiplicação de 8x8 e divisão de 16x16 retiradas de Aplication Notes da própria Microchip.

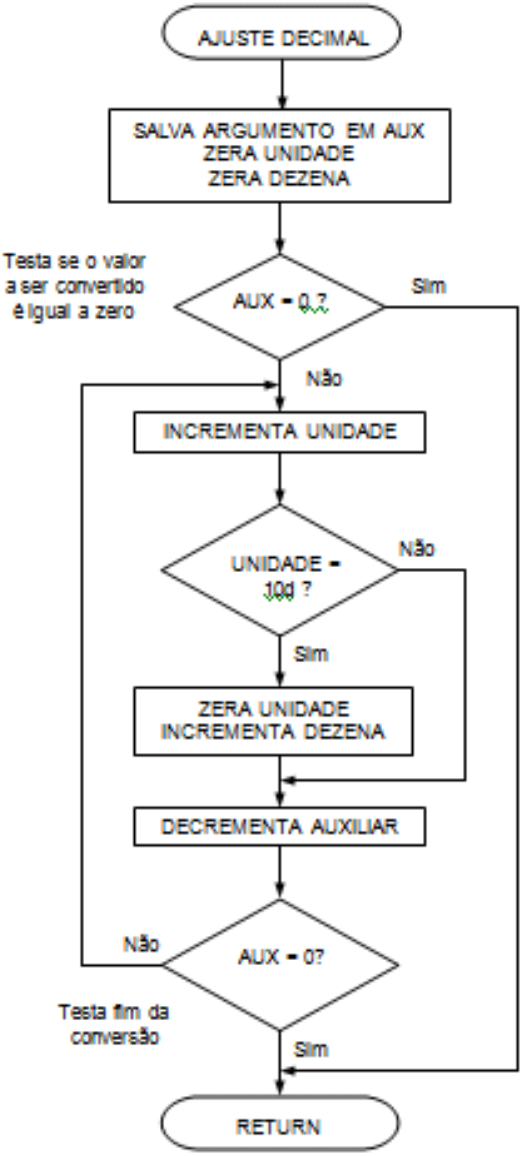
13.3. Esquema Elétrico



13.4. Fluxograma




1
LIMPA WDT
INICIA CONVERSÃO (ADCON0,GO=1)
Não
FIM CONVERSÃO?
Aguarda que o bit GO do registrador ADCON0 seja 0
Sim
MULTIPLICA VALOR DA CONVERSÃO POR 50d
DIVIDE RESULTADO POR 255d
FUNDO DE ESCALA = 5,0V ESTÃO SENDO CONSIDERADOS APENAS OS 8 BITS MAIS SIGNIFICATIVOS DA CONVERSÃO A/D (ADRESH)
AJUSTE DECIMAL
POSICIONA CURSOR DO LCD
CONVERTE DEZENA EM ASCII
ENVIA DEZENA AO DISPLAY
ENVIA "." AO DISPLAY
CONVERTE UNIDADE EM ASCII
ENVIA UNIDADE AO DISPLAY


AJUSTE DECIMAL
SALVA ARGUMENTO EM AUX
ZERA UNIDADE
ZERA DEZENA
Testa se o valor a ser convertido é igual a zero
AUX = 0 ?
Sim
Não
INCREMENTA UNIDADE
UNIDADE = 10d ?
Não
Sim
ZERA UNIDADE
INCREMENTA DEZENA
DECREMENTA AUXILIAR
Não
AUX = 0?
Testa fim da conversão
Sim
RETURN

13.5. Código

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Exemplo de Código para utilização com a McMaster            *
 *                              Exemplo 11                                 *
 *                                                                         *
 *                                MOSAICO                                  *
 *                                                                         *
 *  TEL: (0XX11) 4992-8775                SITE: www.mosaico.com.br         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                          *
 *   DATA : 18/01/2010                                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Descrição geral                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

//Este exemplo foi elaborado para explicar o funcionamento do módulo de
//conversão analógico digital interno do PIC. É convertido o valor analógico
//presente no pino RA0 do microcontrolador, sendo que este valor pode ser
//alterado através do potenciômetro P2 da placa McMaster. O valor da
//conversão A/D é ajustado numa escala de 0 à 5V e mostrado no LCD.
/*
 CONFIGURAÇÂO DOS JUMPERS DA PLACA:

        CAP USB(RC3)              -> ON
        DATA I2C                  -> OFF
        CLOCK I2C                 -> OFF
        COLUNA 2(RB1)             -> ON
        COLUNA 1(RB0)             -> ON
        LINHA 1 / DISPLAY MILHAR(RB4)    -> ON


        LEDS ESPECIAS (RC0,RC1,RC2)      -> ON


*/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           DEFINIÇÃO PIC                                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <p18F4550.h>           // Register definitions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               INCLUDES DAS FUNÇÕES DE PERIFÉRICOS DO PIC                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <pwm.h>                //PWM library functions
#include <adc.h>                //ADC library functions
#include <timers.h>             //Timer library functions
#include <delays.h>             //Delay library functions
#include <i2c.h>                //I2C library functions
#include <stdlib.h>             //Library functions
#include <usart.h>              //USART library functions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Configurações para gravação                      *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
```

```
#pragma config FOSC = HS
#pragma config CPUDIV = OSC1_PLL2
#pragma config WDT = OFF
#pragma config WDTPS = 128
#pragma config LVP = OFF
#pragma config PWRT = ON
#pragma config BOR = OFF
#pragma config BORV = 0
#pragma config PBADEN       = OFF
#pragma config DEBUG        = OFF
#pragma config PLLDIV       = 1
#pragma config USBDIV       = 1
#pragma config FCMEN        = OFF
#pragma config IESO  = OFF
#pragma config VREGEN       = OFF
#pragma config MCLRE        = ON
#pragma config LPT1OSC      = OFF
#pragma config CCP2MX = ON
#pragma config STVREN       = OFF
#pragma config ICPRT        = OFF

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Definição e inicialização das variáveis Globais          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Constantes internas                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de constantes facilita a programação e a manutenção.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                    Declaração dos flags de software                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de flags ajuda na programação e economiza memória RAM.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Definição e inicialização dos port's                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        PROTOTIPAGEM DE FUNÇÕES                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void comando_lcd(unsigned char caracter);
void escreve_lcd(unsigned char caracter);
void limpa_lcd(void);
void inicializa_lcd(void);
void tela_principal(void);

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Constantes internas                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de constantes facilita a programação e a manutenção.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                 Definição e inicialização das variáveis               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.

unsigned int  conversao = 0;    // armazena o resultado da conversão AD
unsigned char conversao1 = 0;   // armazena o resultado da conversão AD
unsigned char unidade;
unsigned char dezena;
unsigned char centena;
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Declaração dos flags de software                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de flags ajuda na programação e economiza memória RAM.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              ENTRADAS                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// As entradas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                               SAÍDAS                                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//As saídas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define     rs      PORTEbits.RE0       // via do lcd que sinaliza recepção de dados ou
comando
#define     enable  PORTEbits.RE1       // enable do lcd

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Rotina que envia um COMANDO para o LCD                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void comando_lcd(unsigned char caracter)
{
    rs = 0;                     // seleciona o envio de um comando
    PORTD = caracter;           // carrega o PORTD com o caracter
    enable = 1 ;                // gera pulso no enable
    Delay10TCYx(1);             // espera 3 microsegundos
    enable = 0;                 // desce o pino de enable
    Delay10TCYx(4);             // espera mínimo 40 microsegundos
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               Rotina que envia um DADO a ser escrito no LCD            *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void escreve_lcd(unsigned char caracter)
{
    rs = 1;                     // seleciona o envio de um comando
    PORTD = caracter;           // carrega o PORTD com o caracter
    enable = 1;                 // gera pulso no enable
    Delay10TCYx(1);             // espera 10 microsegundos
    enable = 0;                 // desce o pino de enable
    Delay10TCYx(4);             // espera mínimo 40 microsegundos
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função para limpar o LCD                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void limpa_lcd(void)
{
    comando_lcd(0x01);          // limpa lcd
    Delay1KTCYx(2);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Inicialização do Display de LCD                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void inicializa_lcd(void)
{
```

```
    comando_lcd(0x30);          // envia comando para inicializar display
    Delay1KTCYx(4);             // espera 4 milisengundos

    comando_lcd(0x30);          // envia comando para inicializar display
    Delay10TCYx(10);            // espera 100 microsengundos

    comando_lcd(0x30);          // envia comando para inicializar display

    comando_lcd(0x38);          // liga o display, sem cursor e sem blink

    limpa_lcd();                // limpa lcd

    comando_lcd(0x0c);          // display sem cursor

    comando_lcd(0x06);          // desloca cursor para a direita
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Tela Principal                                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void tela_principal(void)
{
    comando_lcd(0x83);    // posiciona o cursor na linha 0, coluna 0
    escreve_lcd ('V');
    escreve_lcd ('o');
    escreve_lcd ('l');
    escreve_lcd ('t');
    escreve_lcd ('i');
    escreve_lcd ('m');
    escreve_lcd ('e');
    escreve_lcd ('t');
    escreve_lcd ('r');
    escreve_lcd ('o');    // imprime mensagem no lcd
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Converte BCD                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void converte_bcd(unsigned char aux)
{
    unidade = 0;
    dezena = 0;
    centena = 0;
    if (aux == 0)return;
    while(aux--)
    {
        unidade++;
        if(unidade != 10)continue;
        unidade = 0;
        dezena++;
        if (dezena != 10)continue;
        dezena = 0;
        centena++;
        if (centena != 10)continue;
        centena = 0;
    }
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Configurações do Pic                              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void main ()
{
```

```
    PORTA = 0x00;               //Limpa PORTA
    PORTB = 0x00;               //Limpa PORTB
    PORTC = 0x00;               //Limpa PORTC
    PORTD = 0x00;               //Limpa PORTD
    PORTE = 0x00;               //Limpa PORTE

    LATA = 0x00;                //Limpa PORTA
    LATB = 0x00;                //Limpa PORTB
    LATC = 0x00;                //Limpa PORTC
    LATD = 0x00;                //Limpa PORTD
    LATE = 0x00;                //Limpa PORTE

    TRISA=(0b11101111);         // configuração da direção dos pinos de I/O
    TRISB=(0b11111111);
    TRISC=(0b11111111);
    TRISD=(0b00000000);
    TRISE=(0b00000100);

    //CONFIGURAÇÃO DO AD
 ADCON0 = 0x01;
 ADCON1 = 0x0E;

    inicializa_lcd();           // configura o lcd
    tela_principal();           // imprime a tela principal no LCD

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            Rotina principal                            *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
    while(1)                    // rotina principal
    {
        ClrWdt();
        ConvertADC();           //Inicia conversão AD
        while (BusyADC());      //Aguarda Fim da conversão AD
        conversao = ADRESH;     //lê resultado da conversão AD
        conversao = (conversao * 50);   //faz regra de 3 para converter o valor,
        conversao = (conversao / 255);  //das unidades de AD em Volts.
        conversao1 = conversao;
        converte_bcd(conversao1);
        comando_lcd(0xC5);              //posiciona o cursor na linha 1, coluna 2
        escreve_lcd (dezena + 0x30);    //escreve no display de LCD
        escreve_lcd (',');
        escreve_lcd (unidade + 0x30);   //escreve no display de LCD
        escreve_lcd ('V');
    }

}   // FIM DO PROGRAMA
```

13.6. Dicas e Comentários

Inicialmente notar que toda a estrutura e rotinas utilizadas para a escrita no LCD são as mesmas já aplicadas na experiência anterior.

Observar também que não foi utilizada nenhuma interrupção neste programa. Por isso, o programa permanece parado em um pequeno loop enquanto a conversão não termina. Isto é checado através do bit **ADCON0<GO/DONE>**.

Outra rotina bem interessante que aparece neste sistema é a de conversão de um número qualquer (limitado entre 0 e 99) em dois dígitos separados, facilitando assim a escrita no LCD. Esta rotina devolve os dígitos nas variáveis UNIDADE e DEZENA. Não esquecer que antes de transmitir um valor decimal ao LCD, deve-se converte-lo em um caractere ASCII.

Para facilitar as contas e não utilizarmos números fracionários, a conversão para Volts é feita considerando-se 50 no lugar de 5,0 de forma que ao enviar o valor final ao LCD é simplesmente colocada uma vírgula entre os dois dígitos.

13.7. Exercícios Propostos

1. Simule que a entrada analógica é um sensor de temperatura linear que deve marcar de 10 a 80°C;
2. Altere o exemplo para indicar a tensão entre 0 e 2,50V, utilizando 10 bits de resolução. Para isso, faça uso da tensão de referência externa existente na placa de periféricos (pino RA3).

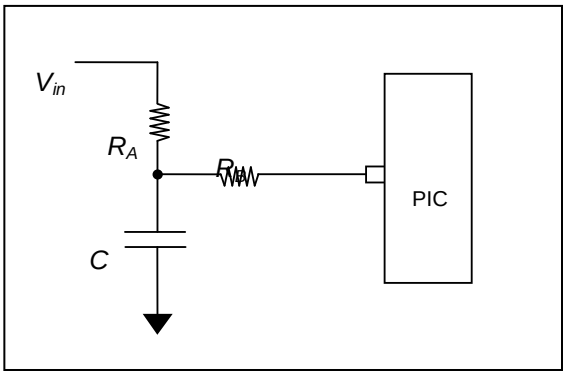
14. Experiência 12 – Conversão A/D via RC

14.1. Objetivo

Nesta experiência será vista uma forma de conversão A/D fundamentada no tempo de carga de um capacitor. A vantagem desta técnica é que o microcontrolador não necessita possuir um conversor A/D interno para estimar o valor de uma variável analógica.

14.2. Descrição

A técnica descrita nesta experiência baseia-se na carga e descarga de um capacitor. A idéia consiste em medir, através do microcontrolador, o tempo de carga de um capacitor num circuito RC. Veja o exemplo de hardware colocado a seguir:



Admitindo-se que o pino do PIC está configurado como entrada, o tempo de carga do capacitor *C* está relacionado com o valor de entrada (V_{in}), do resistor R_A e do próprio capacitor *C*. O resistor R_B não interfere no tempo de carga, pois o pino do PIC está em alta impedância (entrada). Já se o pino do PIC estiver configurado como saída em nível lógico 0 o capacitor tende a se descarregar pelo resistor R_B e carregar pelo resistor R_A. Porém, vamos admitir que o valor do resistor R_B seja muito menor do que o de R_A e, portanto, nesta configuração, podemos desprezar a carga proveniente do resistor R_A e admitir que o capacitor *C* apenas se descarrega através de R_B. Em resumo, o capacitor se carrega através de R_A (pino como entrada) e se descarrega através de R_B (pino como saída em 0) sendo que o tempo de carga/descarga depende do próprio valor do capacitor, da tensão de entrada (V_{in}) e do resistor em questão.

Como funciona então a conversão A/D?

1. O software deve configurar o pino do PIC como saída em 0;
2. Esperar o tempo de descarrega do capacitor *C*. Este tempo deve ser garantido por software conforme os valores dos componentes utilizados;
3. Configurar o pino como entrada, ou seja, permitir a carga do capacitor;
4. Contar o tempo que o capacitor leva para que o PIC entenda nível lógico 1, ou seja, contar o tempo de carga do capacitor;

5. Repetir o processo para uma nova conversão.

O tempo de carga do capacitor será inversamente proporcional à tensão de entrada.

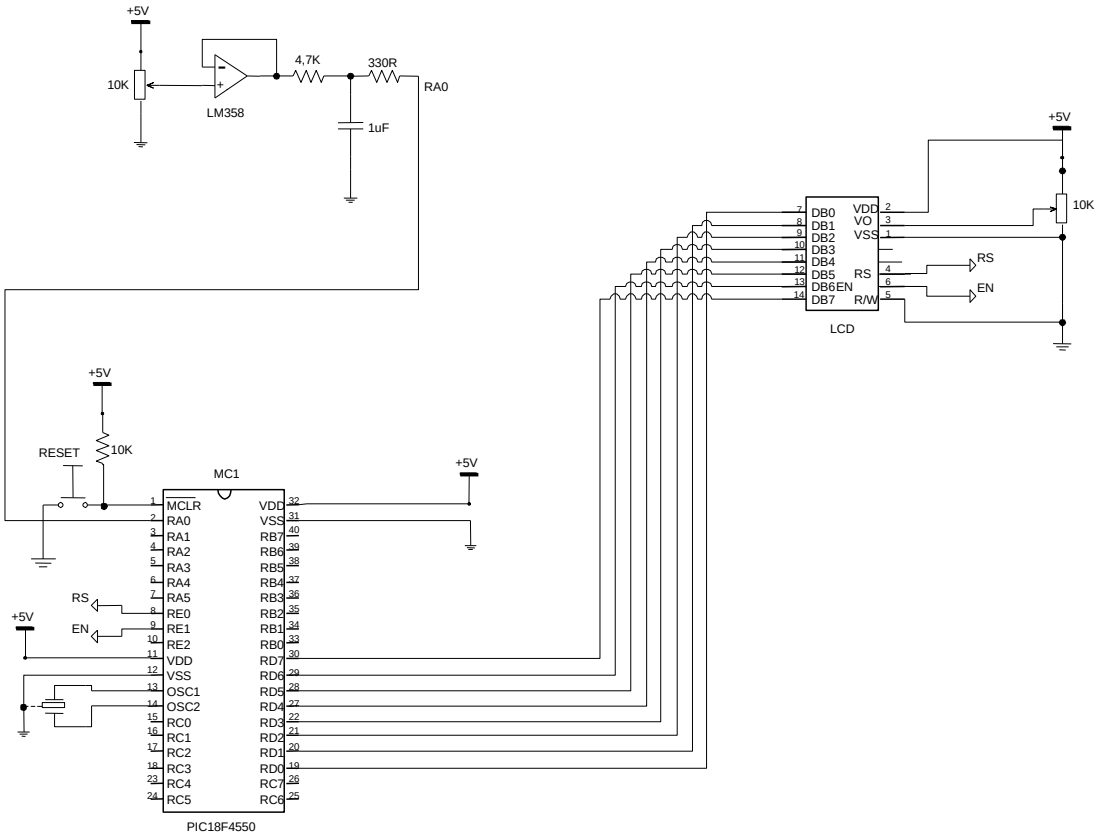
Admitindo-se que a tensão de entrada não varie durante a conversão A/D, o modelo matemático aproximado para a curva de carga do capacitor é

$$V_{cap}(t) = V_{in}\left(1 - e^{-t/R_A C}\right).$$

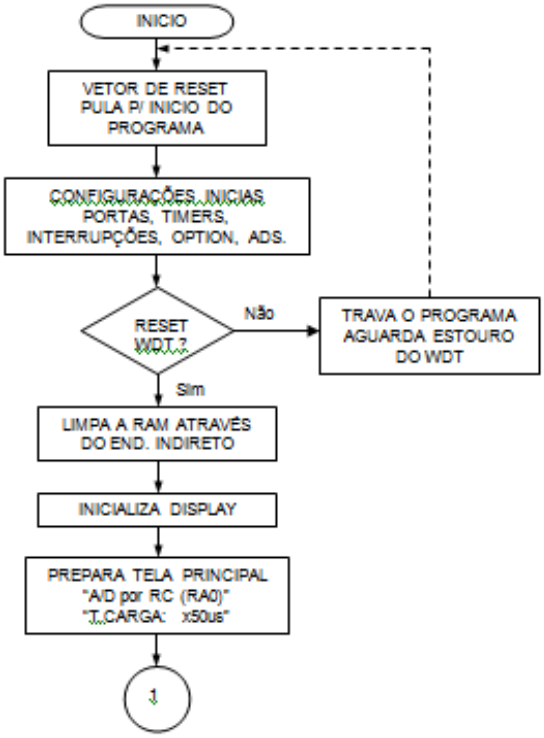
Assim, conhecidos os valores do resistor, do capacitor e do tempo de carga, pode-se estimar a tensão de entrada.

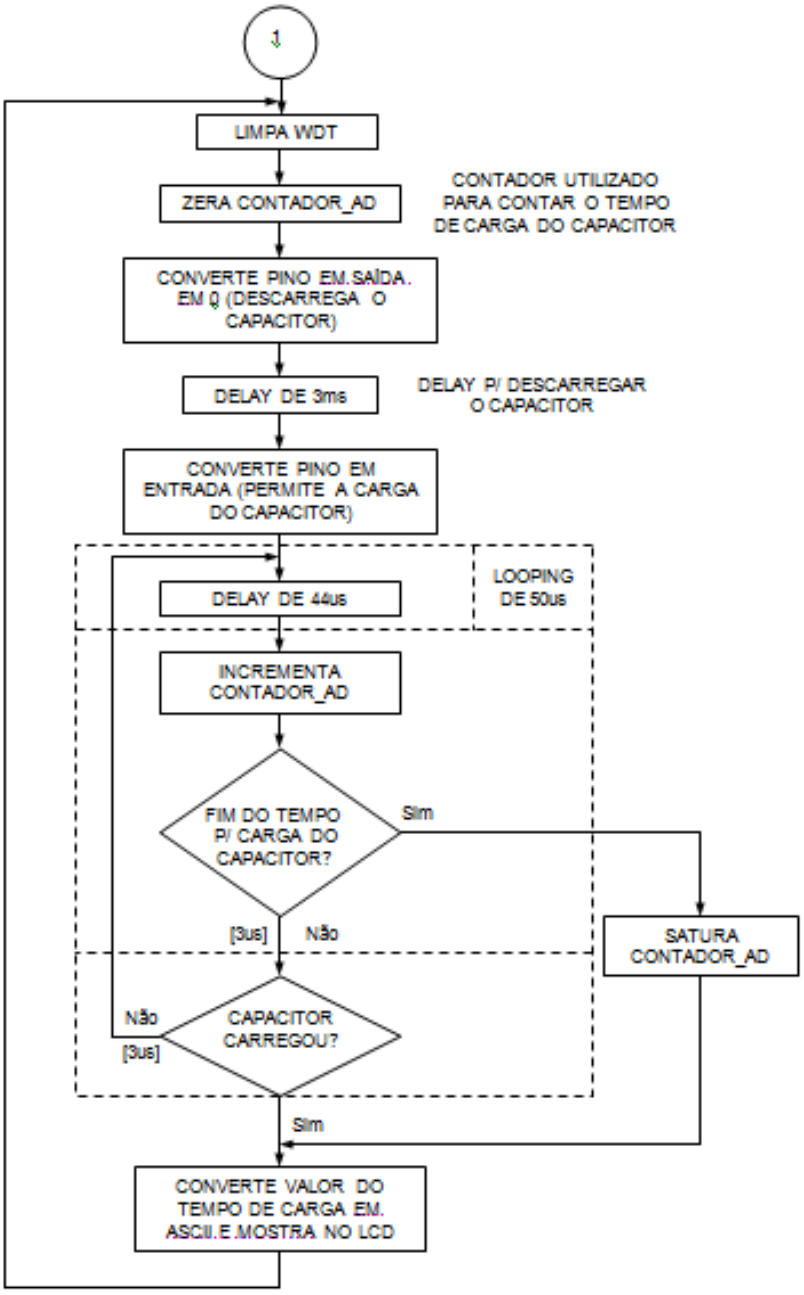
O exemplo de software da experiência calcula o tempo de carga do capacitor e mostra este tempo no display LCD. O software não calcula a tensão de entrada, apenas o tempo de carga do capacitor. O tempo de carga pode ser alterado através do potenciômetro do McMaster.

14.3. Esquema Elétrico



14.4. Fluxograma




1
LIMPA WDT
ZERA CONTADOR_AD
CONTADOR UTILIZADO PARA CONTAR O TEMPO DE CARGA DO CAPACITOR
CONVERTE PINO EM SAÍDA EM 0 (DESCARREGA O CAPACITOR)
DELAY DE 3ms
DELAY P/ DESCARREGAR O CAPACITOR
CONVERTE PINO EM ENTRADA (PERMITE A CARGA DO CAPACITOR)
LOOPING DE 50us
DELAY DE 44us
INCREMENTA CONTADOR_AD
FIM DO TEMPO P/ CARGA DO CAPACITOR?
Sim
[3us]
Não
SATURA CONTADOR_AD
Não
[3us]
CAPACITOR CARREGOU?
Sim
CONVERTE VALOR DO TEMPO DE CARGA EM ASCII E MOSTRA NO LCD

14.5. Código

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Exemplo de Código para utilização com a McMaster            *
 *                              Exemplo 12                                 *
 *                                                                         *
 *                               MOSAICO                                   *
 *                                                                         *
 *  TEL: (0XX11) 4992-8775              SITE: www.mosaico.com.br           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                          *
 *   DATA : 18/01/2010                                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * *  * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Descrição geral                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*
ESTE EXEMPLO FOI ELABORADO PARA EXPLICAR O FUNCIONAMENTO DE UM TIPO DE
CONVERSOR A/D FUNDAMENTADO NO TEMPO DE CARGA DE UM CAPACITOR. O TEMPO DE
CARGA DO CAPACITOR É MOSTRADO NO LCD E É INVERSAMENTE PROPORCIONAL À
TENSÃO APLICADA ATRVÉS DO POTENCIÔMETRO P2. UTILIZE OS JUMPERS (RA5) E DIVISOR
(RESISTOR/ CAPACITOR)
*/
/*
 CONFIGURAÇÂO DOS JUMPERS DA PLACA:

        CAP USB(RC3)              -> ON
        DATA I2C                  -> OFF
        CLOCK I2C                 -> OFF
        COLUNA 2(RB1)             -> ON
        COLUNA 1(RB0)             -> ON
        LINHA 1 / DISPLAY MILHAR(RB4)    -> ON
        LEDS ESPECIAS (RC0,RC1,RC2)      -> ON

*/


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            DEFINIÇÃO PIC                                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <p18F4550.h>            // Register definitions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               INCLUDES DAS FUNÇÕES DE PERIFÉRICOS DO PIC                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <pwm.h>                //PWM library functions
#include <adc.h>                //ADC library functions
#include <timers.h>             //Timer library functions
#include <delays.h>             //Delay library functions
#include <i2c.h>                //I2C library functions
#include <stdlib.h>             //Library functions
#include <usart.h>              //USART library functions


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      Configurações para gravação                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
```

```
#pragma config FOSC = HS
#pragma config CPUDIV = OSC1_PLL2
#pragma config WDT = OFF
#pragma config WDTPS = 128
#pragma config LVP = OFF
#pragma config PWRT = ON
#pragma config BOR = OFF
#pragma config BORV = 0
#pragma config PBADEN = OFF
#pragma config DEBUG = OFF
#pragma config PLLDIV = 1
#pragma config USBDIV = 1
#pragma config FCMEN = OFF
#pragma config IESO = OFF
#pragma config VREGEN = OFF
#pragma config MCLRE = ON
#pragma config LPT1OSC = OFF
#pragma config CCP2MX = ON
#pragma config STVREN = OFF
#pragma config ICPRT = OFF

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Definição e inicialização das variáveis Globais          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.
unsigned char contador_ad;
unsigned char aux;

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Constantes internas                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de constantes facilita a programação e a manutenção.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Declaração dos flags de software                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de flags ajuda na programação e economiza memória RAM.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Definição e inicialização dos port's                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       PROTOTIPAGEM DE FUNÇÕES                         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void comando_lcd(unsigned char caracter);
void escreve_lcd(unsigned char caracter);
void limpa_lcd(void);
void inicializa_lcd(void);
void escreve_frase(const rom char *frase);
void tela_principal(void);

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              ENTRADAS                                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// As entradas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define     PINO_AD     TRISAbits.TRISA5     //PINO P/ LEITURA DO RC
                                             //0 -> FORÇA A DESCARGA DO CAPACITOR
                                             //1 -> LIBERA A CARGA DO CAPACITOR

#define     CAD         PORTAbits.RA5        //PINO P/ LEITURA DO CONV. A/D
                                             //0 -> CAPACITOR DESCARREGADO
                                             //1 -> CAPACITOR CARREGADO
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              SAÍDAS                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//As saídas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define rs        PORTEbits.RE0    // via do lcd que sinaliza recepção de dados ou
comando
#define enable    PORTEbits.RE1    // enable do lcd

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               Rotina que envia um COMANDO para o LCD                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void comando_lcd(unsigned char caracter)
{
    rs = 0;                         // seleciona o envio de um comando
    PORTD = caracter;               // carrega o PORTD com o caracter
    enable = 1 ;                    // gera pulso no enable
    Delay10TCYx(1);                 // espera 10 microsegundos
    enable = 0;                     // desce o pino de enable
    Delay10TCYx(4);                 // espera mínimo 40 microsegundos
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *            Rotina que envia um DADO a ser escrito no LCD              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void escreve_lcd(unsigned char caracter)
{
    rs = 1;                     // seleciona o envio de um comando
    PORTD = caracter;           // carrega o PORTD com o caracter
    enable = 1;                 // gera pulso no enable
    Delay10TCYx(1);             // espera 10 microsegundos
    enable = 0;                 // desce o pino de enable
    Delay10TCYx(4);             // espera mínimo 40 microsegundos
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função para limpar o LCD                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void limpa_lcd(void)
{
    comando_lcd(0x01);            // limpa lcd
    Delay1KTCYx(2);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Inicialização do Display de LCD                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void inicializa_lcd(void)
{
    Delay1KTCYx(15);            // espera 15 milisengundos
    comando_lcd(0x30);          // envia comando para inicializar display
    Delay1KTCYx(4);             // espera 4 milisengundos

    comando_lcd(0x30);          // envia comando para inicializar display
    Delay10TCYx(10);            // espera 100 microsengundos

    comando_lcd(0x30);          // envia comando para inicializar display

    comando_lcd(0x38);          // liga o display, sem cursor e sem blink
```

```
    limpa_lcd();                // limpa lcd

    comando_lcd(0x0c);          // display sem cursor

    comando_lcd(0x06);          // desloca cursor para a direita
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *            Função para escrever uma frase no LCD                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void escreve_frase(const rom char *frase)
{
    do
    {
        escreve_lcd(*frase);
    }while(*++frase);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Tela Principal                                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void tela_principal(void)
{
    comando_lcd(0x80);            // posiciona o cursor na linha 0, coluna 0
    escreve_frase("A/D por RC (RA5)");
    comando_lcd(0xC0);            // posiciona o cursor na linha 1, coluna 0
    escreve_frase("T.CARGA:   x50us");
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função Principal                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void main()
{
    PORTA = 0x00;               //Limpa PORTA
    PORTB = 0x00;               //Limpa PORTB
    PORTC = 0x00;               //Limpa PORTC
    PORTD = 0x00;               //Limpa PORTD
    PORTE = 0x00;               //Limpa PORTE

    LATA = 0x00;                //Limpa PORTA
    LATB = 0x00;                //Limpa PORTB
    LATC = 0x00;                //Limpa PORTC
    LATD = 0x00;                //Limpa PORTD
    LATE = 0x00;                //Limpa PORTE


    TRISA = 0b11101111;                 //CONFIG DIREÇÃO DOS PINOS PORTA0
    TRISB = 0b00001111;                 //CONFIG DIREÇÃO DOS PINOS PORTB
    TRISC = 0b11111111;                 //CONFIG DIREÇÃO DOS PINOS PORTC
    TRISD = 0b00000000;                 //CONFIG DIREÇÃO DOS PINOS PORTD
    TRISE = 0b00000100;                 //CONFIG DIREÇÃO DOS PINOS PORTE

    ADCON1 = 0b00001111;                //DESLIGA CONVERSORES A/D

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Inicialização do Sistema                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

    inicializa_lcd();            // configura o lcd
    tela_principal();            // imprime a tela principal no LCD

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
 *                              Rotina Principal                              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
    while(1)
    {
        ClrWdt();               //LIMPA O WDT

        contador_ad = 0;        //REINICIA CONTADOR AD

        PINO_AD = 0;            //CONFIGURA PINO DO AD COMO SAÍDA
        CAD = 0;                //FORÇA PINO NO AD EM 0 PARA DECARGA DO CAPACITOR
        Delay1KTCYx(3);         //DELAY DE 3ms PARA GARANTIR A DESCARGA DO CAPACITOR
        PINO_AD = 1;            //CONFIGURA PINO DO AD COMO ENTRADA

        while(1)
        {
            Delay10TCYx(5);     //DELAY DE 50us
            contador_ad++;      //INCREMENTA CONTADOR DO AD
            if(!contador_ad)    //CONTADOR = 0?
            {
                contador_ad = 255;    //CONTADOR = 255 (SATURA AD)
                break;                //SAI DO LAÇO
            }
            if(CAD) break;            //SE PINO CAD = 1, SAI DO LAÇO
        }
        comando_lcd(0xC9);            //POSICIONA CURSOS NA LINHA 1, COLUNA 9

        aux = contador_ad >> 4;        //CONVERTE CONTADOR EM HEXADECIMAL
        if(aux < 10) aux = aux + 0x30;
        else aux = aux + 0x37;
        escreve_lcd(aux);

        aux = contador_ad & 0b00001111;
        if(aux < 10) aux = aux + 0x30;
        else aux = aux + 0x37;
        escreve_lcd(aux);
    }
}
```

14.6. Dicas e Comentários

Este tipo de conversor A/D não apresenta uma boa precisão, além de apresentar uma série de inconvenientes:

- Note que a tensão sobre o capacitor não varia linearmente no tempo e, portanto, este tipo de conversor A/D não é linear;
- O valor da conversão, ou seja, o tempo de carga do capacitor está sujeito às variações dos componentes envolvidos.
- Normalmente o capacitor é muito suscetível a variações térmicas;
- A tensão de entrada deve ser suficientemente alta para que o PIC entenda nível lógico 1, por isso este conversor não funciona no range completo de 0 a 5V;
- O valor de tensão necessário para que o PIC entenda nível lógico 1 pode variar em função da pastilha, da tensão da fonte (alimentação do PIC) e do tipo de pino (TTL/ST).

Como dica, podemos sugerir:

- Utilizar R_B pelo menos 10 vezes menor que R_A;
- Não utilizar capacitores maiores do que $1\mu F$;
- Dar preferência ao uso de capacitores de tântalo ou cerâmico;
- Não discretizar mais do que oito níveis.

14.7. Exercícios Propostos

1. Admitindo que a tensão de entrada varia entre 0 e 5V, estimar através do tempo de carga do capacitor se a tensão de entrada encontra-se abaixo de 1,25V, entre 1,25V e 2,5V, entre 2,5V e 3,75V ou acima de 3,75V.

15. Experiência 13 – Leitura de jumpers via RC

15.1. Objetivo

O objetivo desta experiência é aplicar a técnica apresentada na aula anterior para viabilizar a leitura de jumpers através de um único pino do microcontrolador.

15.2. Descrição

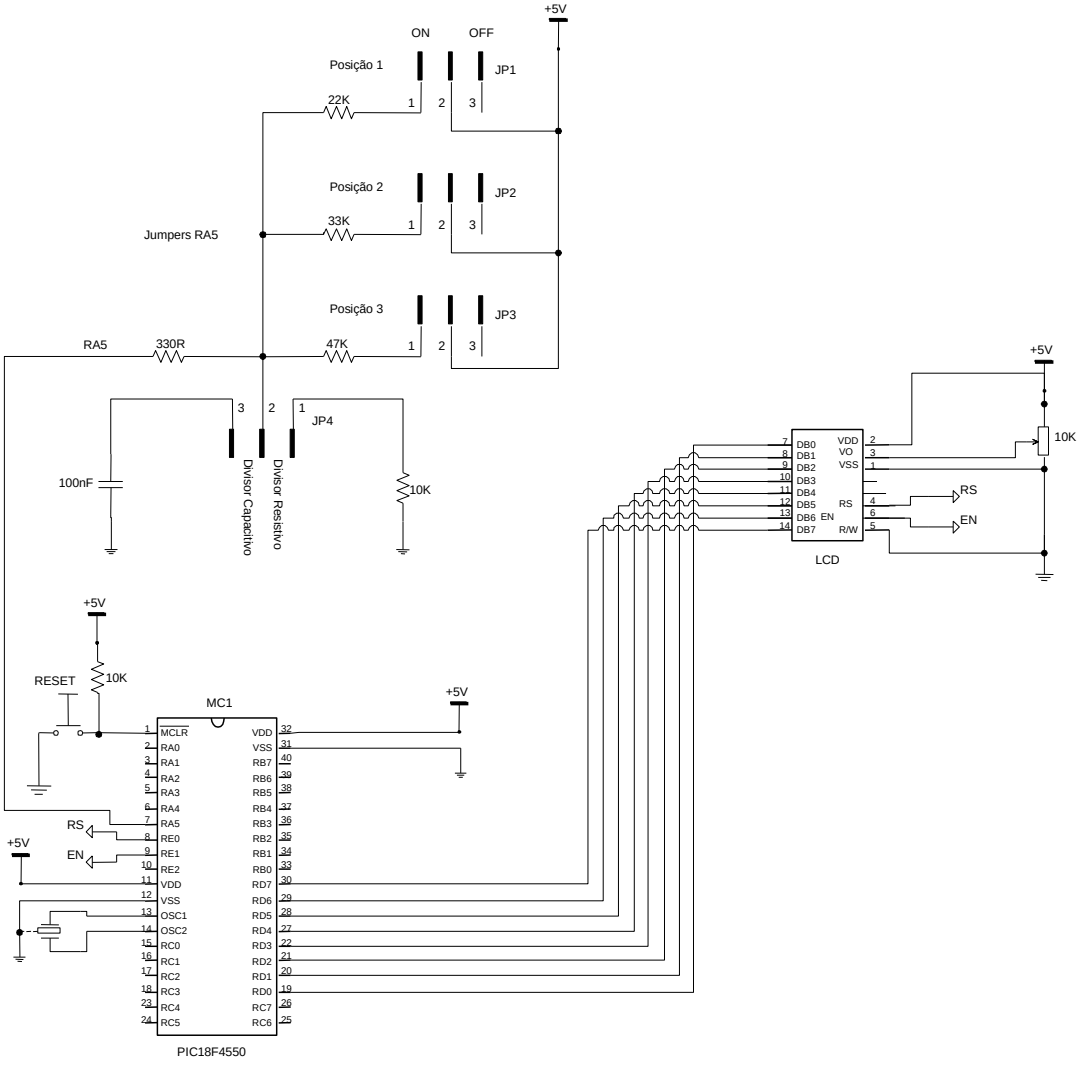
Assim como na experiência anterior, este exemplo foi elaborado utilizando a técnica de conversão A/D através de um circuito RC.

Diferente do caso anterior, a tensão de entrada nesta experiência é fixa e o que varia é o valor do resistor que fornece a carga do capacitor. Utilizando os jumpers A, B e C do McMaster pode-se alterar o valor do resistor de carga e conseqüentemente o tempo de carga do capacitor. Como são 3 jumpers, existem até 8 combinações possíveis que alteram o tempo de carga do capacitor.

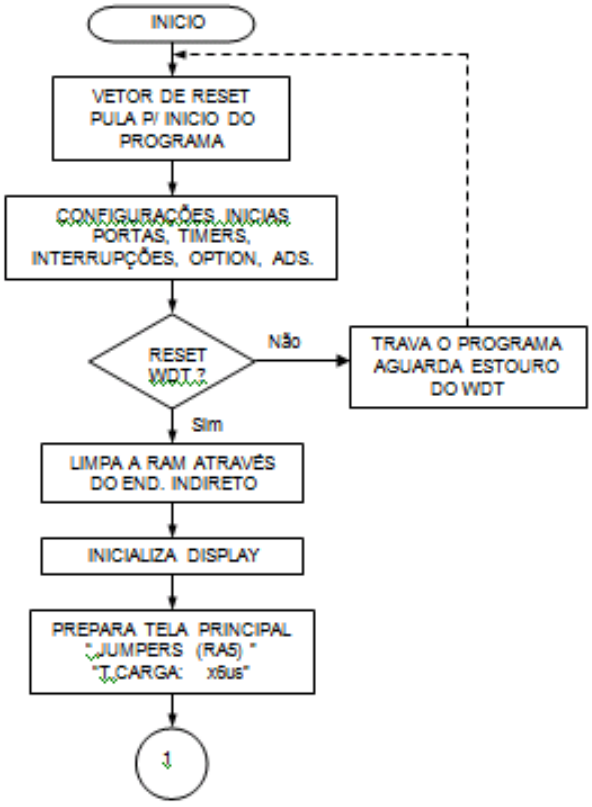
O quarto jumper deve estar selecionado na posição resistor/capacitor para que o exemplo possa ser utilizado.

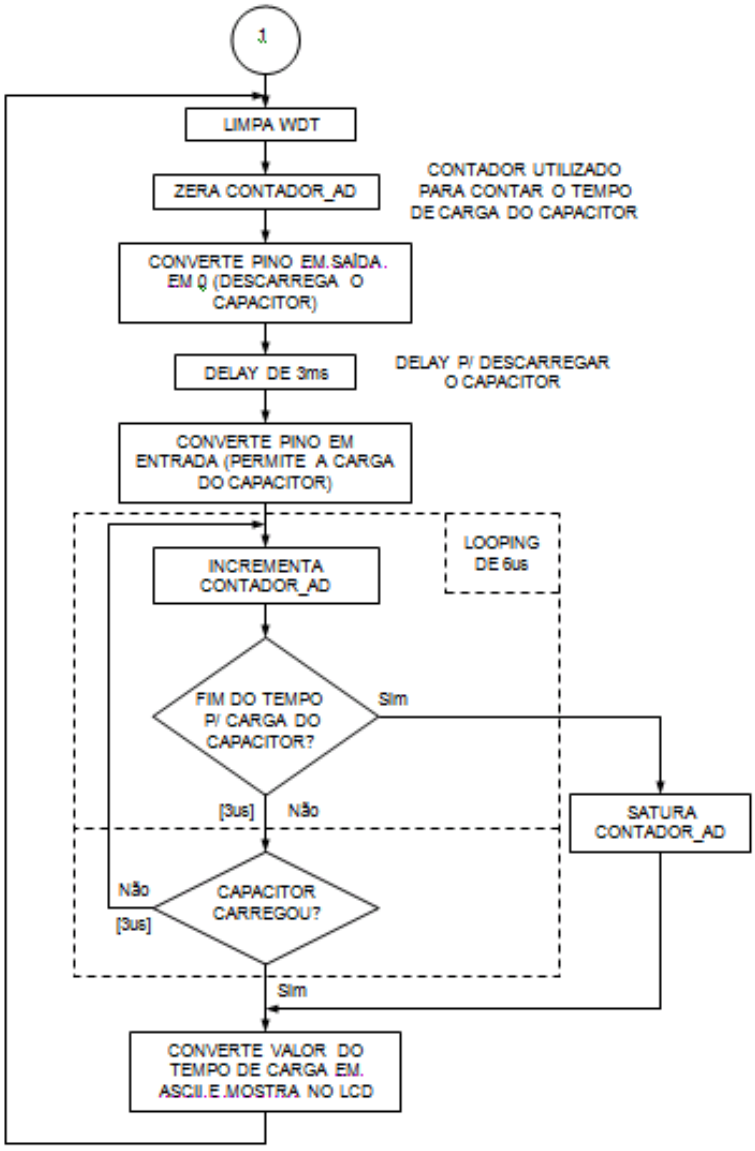
O software calcula o tempo de carga do capacitor e mostra o valor no display LCD. O interessante desta técnica é que ao invés de utilizar 3 pinos do microcontrolador, um para cada jumper, ela utiliza apenas um.

15.3. Esquema Elétrico



15.4. Fluxograma




1
LIMPA WDT
ZERA CONTADOR_AD
CONTADOR UTILIZADO PARA CONTAR O TEMPO DE CARGA DO CAPACITOR
CONVERTE PINO EM SAÍDA. EM 0 (DESCARREGA O CAPACITOR)
DELAY DE 3ms
DELAY P/ DESCARREGAR O CAPACITOR
CONVERTE PINO EM ENTRADA (PERMITE A CARGA DO CAPACITOR)
LOOPING DE 6us
INCREMENTA CONTADOR_AD
FIM DO TEMPO P/ CARGA DO CAPACITOR?
Sim
[3us]
Não
SATURA CONTADOR_AD
Não
[3us]
CAPACITOR CARREGOU?
Sim
CONVERTE VALOR DO TEMPO DE CARGA EM ASCII.E.MOSTRA NO LCD

15.5. Código

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Exemplo de Código para utilização com a McMaster           *
 *                                Exemplo 13                               *
 *                                                                         *
 *                                 MOSAICO                                 *
 *                                                                         *
 *  TEL: (0XX11) 4992-8775                SITE: www.mosaico.com.br         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                          *
 *   DATA : 18/01/2010                                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * *  * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            Descrição geral                              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*
ESTE EXEMPLO FOI ELABORADO PARA EXPLICAR O FUNCIONAMENTO DE UM TIPO DE
CONVERSOR A/D FUNDAMENTADO NO TEMPO DE CARGA DE UM CAPACITOR. O TEMPO DE
CARGA PODE SER ALTERADO UTILIZANDO OS JUMPERS LIGADOS AO PINO RA5.
COM SÃO TRÊS JUMPERS EXISTEM 8 COMBINAÇÕES POSSÍVEIS QUE ALTERAM O TEMPO
DE CARGA DO CAPACITOR. O QUARTO JUMPER DEVE ESTAR SELECIONADO NA POSIÇÃO
RESISTOR/CAPACITOR PARA QUE O EXEMPLO POSSA SER UTILIZADO.
*/
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           DEFINIÇÃO PIC                                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <p18F4550.h>             // Register definitions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               INCLUDES DAS FUNÇÕES DE PERIFÉRICOS DO PIC                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <pwm.h>                //PWM library functions
#include <adc.h>                //ADC library functions
#include <timers.h>             //Timer library functions
#include <delays.h>             //Delay library functions
#include <i2c.h>                //I2C library functions
#include <stdlib.h>             //Library functions
#include <usart.h>              //USART library functions


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Configurações para gravação                      *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#pragma config FOSC = HS
#pragma config CPUDIV = OSC1_PLL2
#pragma config WDT = OFF
#pragma config WDTPS = 128
#pragma config LVP = OFF
#pragma config PWRT = ON
#pragma config BOR = OFF
#pragma config BORV = 0
#pragma config PBADEN = OFF
#pragma config DEBUG = OFF
#pragma config PLLDIV = 1
#pragma config USBDIV = 1
```

```
#pragma config FCMEN = OFF
#pragma config IESO = OFF
#pragma config VREGEN = OFF
#pragma config MCLRE = ON
#pragma config LPT1OSC = OFF
#pragma config CCP2MX = ON
#pragma config STVREN = OFF
#pragma config ICPRT = OFF

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Definição e inicialização das variáveis Globais          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.
unsigned char contador_ad;
unsigned char aux;

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Constantes internas                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de constantes facilita a programação e a manutenção.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Declaração dos flags de software                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de flags ajuda na programação e economiza memória RAM.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                 Definição e inicialização dos port's                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                     PROTOTIPAGEM DE FUNÇÕES                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void comando_lcd(unsigned char caracter);
void escreve_lcd(unsigned char caracter);
void limpa_lcd(void);
void inicializa_lcd(void);
void escreve_frase(const rom char *frase);
void tela_principal(void);

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                             ENTRADAS                                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// As entradas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define    PINO_AD    TRISAbits.TRISA5    //PINO P/ LEITURA DO RC
                                          //0 -> FORÇA A DESCARGA DO CAPACITOR
                                          //1 -> LIBERA A CARGA DO CAPACITOR

#define    CAD        PORTAbits.RA5       //PINO P/ LEITURA DO CONV. A/D
                                          //0 -> CAPACITOR DESCARREGADO
                                          //1 -> CAPACITOR CARREGADO

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                             SAÍDAS                                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//As saídas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define rs        PORTEbits.RE0    // via do lcd que sinaliza recepção de dados ou
comando
#define enable    PORTEbits.RE1    // enable do lcd

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               Rotina que envia um COMANDO para o LCD                  *
```

```
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void comando_lcd(unsigned char caracter)
{
    rs = 0;                         // seleciona o envio de um comando
    PORTD = caracter;               // carrega o PORTD com o caracter
    enable = 1 ;                    // gera pulso no enable
    Delay10TCYx(1);                 // espera 10 microsegundos
    enable = 0;                     // desce o pino de enable
    Delay10TCYx(4);                 // espera mínimo 40 microsegundos
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Rotina que envia um DADO a ser escrito no LCD                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void escreve_lcd(unsigned char caracter)
{
    rs = 1;                     // seleciona o envio de um comando
    PORTD = caracter;           // carrega o PORTD com o caracter
    enable = 1;                 // gera pulso no enable
    Delay10TCYx(1);             // espera 10 microsegundos
    enable = 0;                 // desce o pino de enable
    Delay10TCYx(4);             // espera mínimo 40 microsegundos
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            Função para limpar o LCD                         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void limpa_lcd(void)
{
    comando_lcd(0x01);            // limpa lcd
    Delay1KTCYx(2);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Inicialização do Display de LCD                            *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void inicializa_lcd(void)
{
    Delay1KTCYx(15);            // espera 15 milisengundos
    comando_lcd(0x30);          // envia comando para inicializar display
    Delay1KTCYx(4);             // espera 4 milisengundos

    comando_lcd(0x30);          // envia comando para inicializar display
    Delay10TCYx(10);            // espera 100 microsengundos

    comando_lcd(0x30);          // envia comando para inicializar display

    comando_lcd(0x38);          // liga o display, sem cursor e sem blink

    limpa_lcd();                // limpa lcd

    comando_lcd(0x0c);          // display sem cursor

    comando_lcd(0x06);          // desloca cursor para a direita
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Função para escrever uma frase no LCD                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void escreve_frase(const rom char *frase)
{
```

```
    do
    {
        escreve_lcd(*frase);
    }while(*++frase);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Tela Principal                                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void tela_principal(void)
{
    comando_lcd(0x81);            // posiciona o cursor na linha 0, coluna 1
    escreve_frase("JUMPER  (RA5)");
    comando_lcd(0xC0);            // posiciona o cursor na linha 1, coluna 0
    escreve_frase("T.CARGA:   x50us");
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função Principal                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void main()
{
    PORTA = 0x00;               //Limpa PORTA
    PORTB = 0x00;               //Limpa PORTB
    PORTC = 0x00;               //Limpa PORTC
    PORTD = 0x00;               //Limpa PORTD
    PORTE = 0x00;               //Limpa PORTE

    LATA = 0x00;                //Limpa PORTA
    LATB = 0x00;                //Limpa PORTB
    LATC = 0x00;                //Limpa PORTC
    LATD = 0x00;                //Limpa PORTD
    LATE = 0x00;                //Limpa PORTE


    TRISA = 0b11111111;                 //CONFIG DIREÇÃO DOS PINOS PORTA0
    TRISB = 0b00001111;                 //CONFIG DIREÇÃO DOS PINOS PORTB
    TRISC = 0b11111111;                 //CONFIG DIREÇÃO DOS PINOS PORTC
    TRISD = 0b00000000;                 //CONFIG DIREÇÃO DOS PINOS PORTD
    TRISE = 0b00000100;                 //CONFIG DIREÇÃO DOS PINOS PORTE

    ADCON1 = 0b00001111;                //DESLIGA CONVERSORES A/D

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Inicialização do Sistema                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

    inicializa_lcd();            // configura o lcd
    tela_principal();            // imprime a tela principal no LCD

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Rotina Principal                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
    while(1)
    {
        ClrWdt();               //LIMPA O WDT

        contador_ad = 0;        //REINICIA CONTADOR AD

        PINO_AD = 0;            //CONFIGURA PINO DO AD COMO SAÍDA
        CAD = 0;                //FORÇA PINO NO AD EM 0 PARA DECARGA DO CAPACITOR
        Delay1KTCYx(3);         //DELAY DE 3ms PARA GARANTIR A DESCARGA DO CAPACITOR
        PINO_AD = 1;            //CONFIGURA PINO DO AD COMO ENTRADA
```

```
        while(1)
        {
            Delay10TCYx(5);      //DELAY DE 50us
            contador_ad++;       //INCREMENTA CONTADOR DO AD
            if(!contador_ad)     //CONTADOR = 0?
            {
                contador_ad = 255;     //CONTADOR = 255 (SATURA AD)
                break;                 //SAI DO LAÇO
            }
            if(CAD) break;             //SE PINO CAD = 1, SAI DO LAÇO
        }
        comando_lcd(0xC9);             //POSICIONA CURSOS NA LINHA 1, COLUNA 9

        aux = contador_ad >> 4;         //CONVERTE CONTADOR EM HEXADECIMAL
        if(aux < 10) aux = aux + 0x30;
        else aux = aux + 0x37;
        escreve_lcd(aux);

        aux = contador_ad & 0b00001111;
        if(aux < 10) aux = aux + 0x30;
        else aux = aux + 0x37;
        escreve_lcd(aux);
    }
}
```

15.6. Dicas e Comentários

As dicas e comentários desta experiência são os mesmos da experiência anterior.

15.7. Exercícios Propostos

1. Através do conhecimento do tempo de carga do capacitor, estime a real posição dos jumpers A, B e C do sistema.

16. Experiência 14 – Modulo PWM

16.1. Objetivo

O objetivo desta experiência é ensinar ao aluno como utilizar o módulo PWM do microcontrolador PIC18F4550

16.2. Descrição

Este PIC possui 2 canais de PWMs (CCP1 e CCP2), cada um com resolução máxima de 10 bits. Isto significa que o duty cycle poderá ser regulado de 0 a 100% com uma resolução máxima de 1024 pontos. No entanto, dependendo da configuração adotada, esta resolução não será atingida.

O período do PWM é controlado diretamente pelo TMR2, através do registrador **PR2**. Como já foi visto, sempre que TMR2 = **PR2**, o timer é zerado e neste momento, um novo período do PWM é iniciado. Desta forma, pode-se definir o período e a freqüência do PWM pelas seguintes fórmulas:

$$T = [(\mathbf{PR2}) + 1] \times 4 \times Tosc \times (\text{Prescale do TMR2})$$

$$PWM_{Freq} = 1 / T$$

O duty cycle normalmente é definido em porcentagem, porém, o PIC não define o valor do duty cycle e sim o tempo do pulso em nível alto. Desta forma, o tempo do pulso pode ser calculado por:

$$tp = \mathbf{CCPRxL:CCPxCON<CCPxX:CCPxY>} \times Tosc \times (\text{Prescale do TMR2})$$

Repare que a largura do pulso é ajustada em dois registradores: **CCPRxL** que armazena os 8 bits mais significativos e **CCPxCON** que armazena os dois bits menos significativos. Assim, obtêm-se os 10 bits que controlam o duty cycle do PWM.

Apesar do PIC não definir o duty cycle, ele pode ser calculado dividindo o tempo do pulso em nível alto pelo período total do PWM.

$$\frac{t_p}{T} = \frac{\textbf{CCPRxL:CCPxCON<DCxB1:DCxB0>} \times T_{OSC} \times (\text{Prescale do TMR2})}{[(\textbf{PR2}) + 1] \times 4 \times T_{OSC} \times (\text{Prescale do TMR2})}$$

$$\frac{t_p}{T} = \frac{\textbf{CCPRxL:CCPxCON<DCxB1:DCxB0>}}{[(\textbf{PR2}) + 1] \times 4}$$

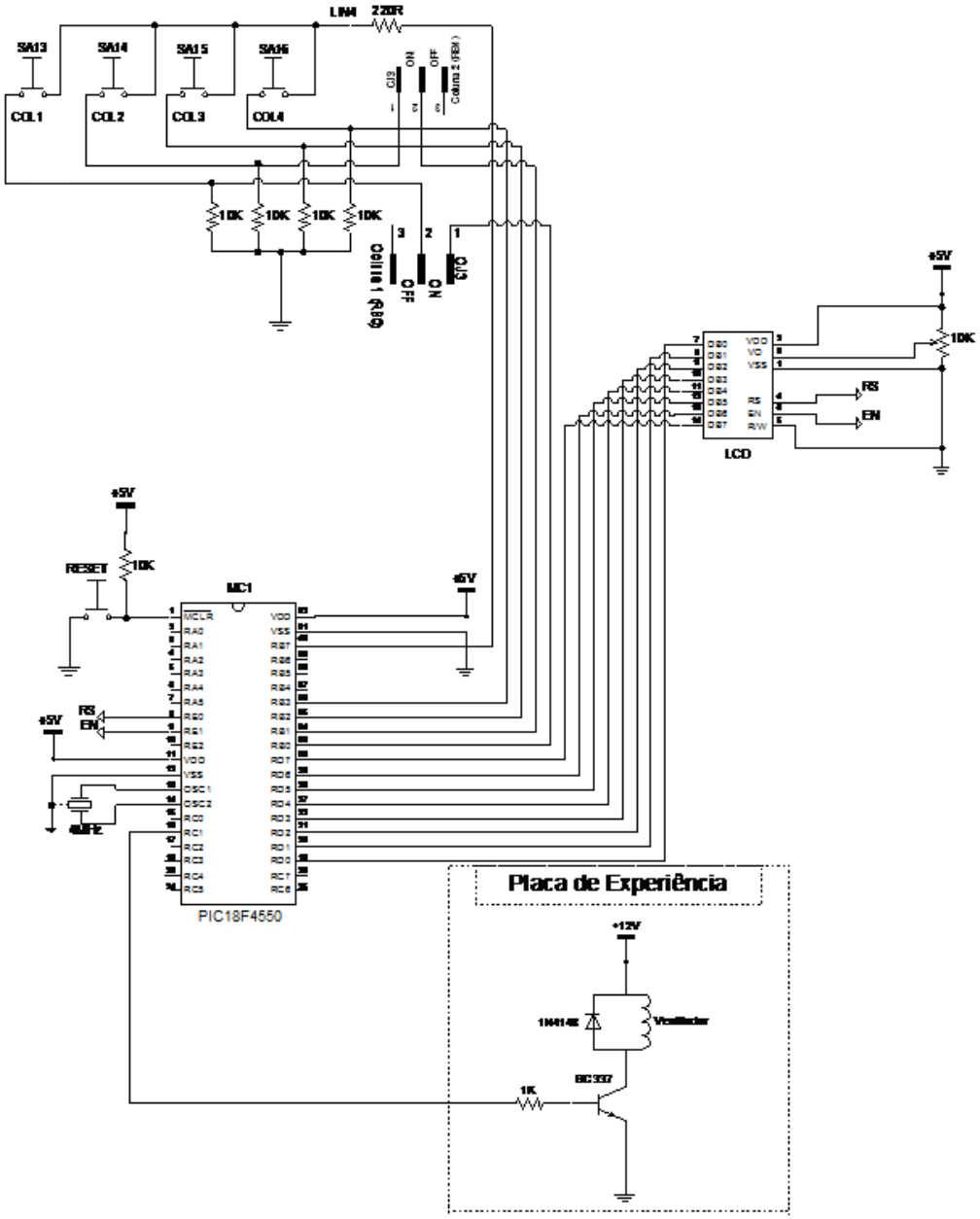
Verifica-se então que apesar do período e o do tempo de pulso dependerem do cristal (Tosc) e do ajuste do prescale do TMR2, o duty cycle depende única e exclusivamente dos valores ajustados nos registradores **PR2**, **CCPRxL** e **CCPxCON** (bits 5 e 4). Veja que o registrador **PR2** (8 bits) que controla o período do PWM é multiplicado por 4 para poder igualar-se aos 10 bits que controlam o duty cycle. É justamente este o problema da resolução máxima atingida. Se o registrador **PR2** for ajustado com um valor menor que 8 bits, ou seja, menor do que 255, serão necessários menos do que 10 bits para atingir um PWM com 100% de duty cycle. Portanto, o número de pontos para ajuste do duty cycle é 4 vezes maior do que o valor ajustado em (**PR2**+1). Em termos de bits, podemos dizer que a resolução do duty cycle é 2 bits maior do que o número de bits que formam o valor ajustado em **PR2**. Repare também que caso **PR2** seja ajustado com 255 nunca será atingido um duty cycle de 100%, pois o período atingirá o valor máximo de 1024 ([**PR2**+1]x4) enquanto o tempo do pulso em nível alto (**<DCxB9:DCxB0>**) será no máximo 1023.

No software da experiência ativou-se a saída do módulo CCP2 para controlar a rotação do ventilador que está ligado ao pino **RC1**. O registrador **PR2** foi ajustado com valor máximo, ou seja, 255 e o prescale do timer foi configurado para 1:16. Com isso a freqüência do PWM ficou em 244,14Hz ($PWM_{Período}$ = 4,096ms), considerando-se que o microcontrolador do sistema McMaster está trabalhando a 4MHz.

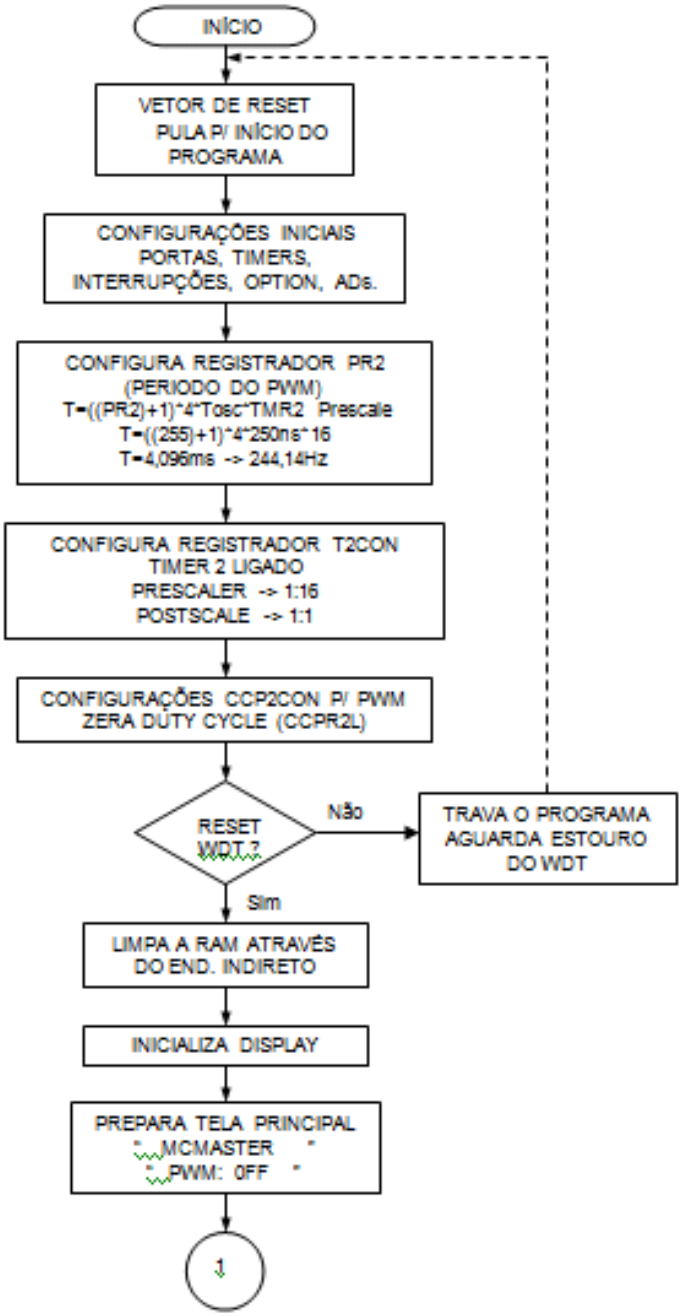
As teclas da linha 4 foram habilitadas e receberam as seguintes funções:

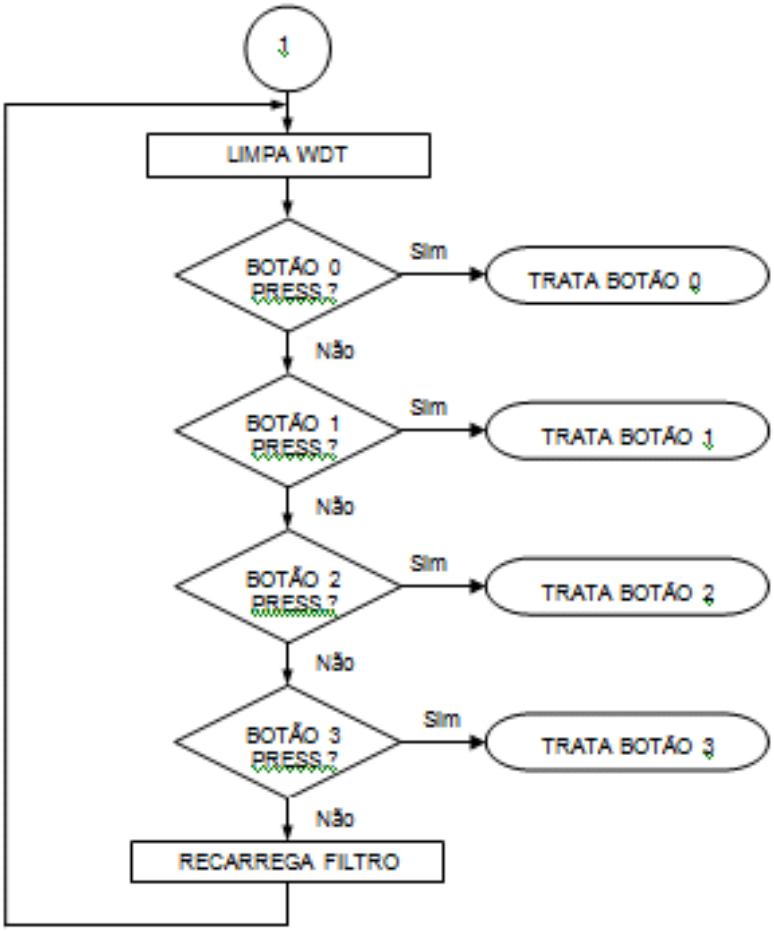
Coluna	Duty Cycle
1	0%
2	50%
3	75%
4	100%

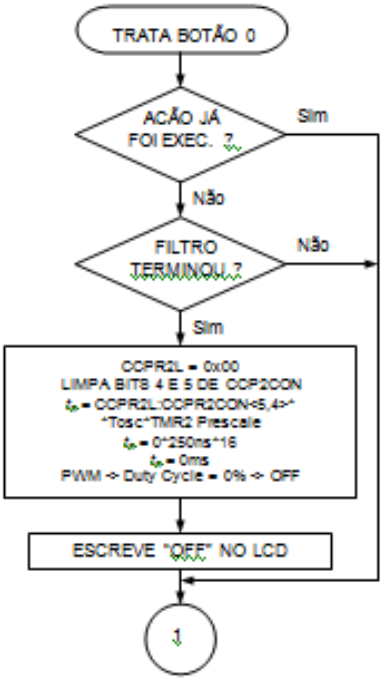
16.3. Esquema Elétrico

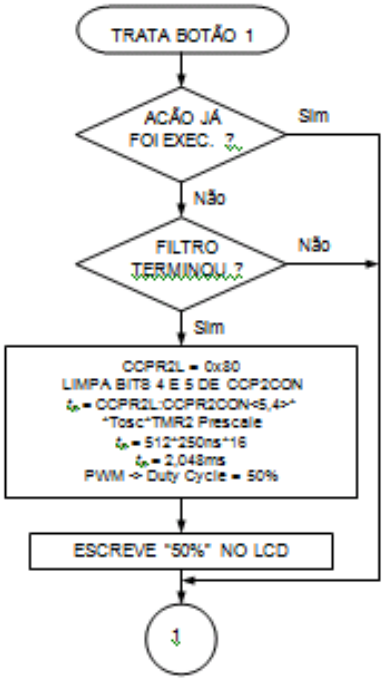


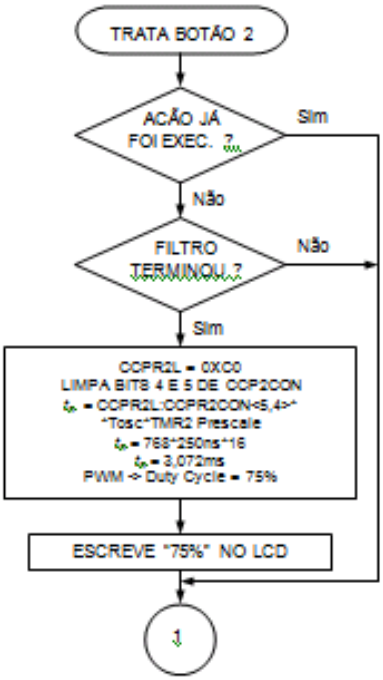
16.4. Fluxograma

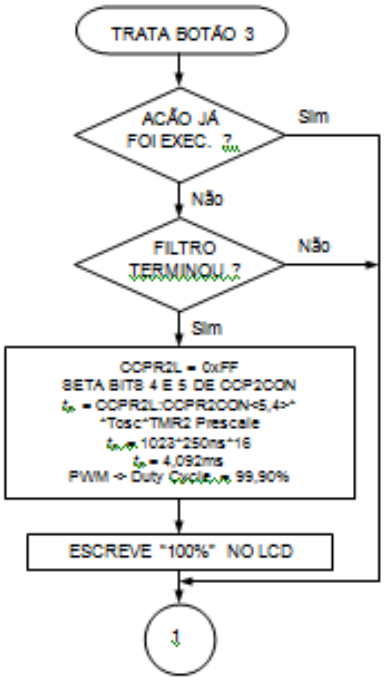



1
LIMPA WDT
BOTÃO 0 PRESS.?
Sim
TRATA BOTÃO 0
Não
BOTÃO 1 PRESS.?
Sim
TRATA BOTÃO 1
Não
BOTÃO 2 PRESS.?
Sim
TRATA BOTÃO 2
Não
BOTÃO 3 PRESS.?
Sim
TRATA BOTÃO 3
Não
RECARREGA FILTRO


TRATA BOTÃO 0
AÇÃO JÁ FOI EXEC. ?
Sim
Não
FILTRO TERMINOU ?
Não
Sim
CCPR2L = 0x00
LIMPA BITS 4 E 5 DE CCP2CON
t_{on} = CCPR2L:CCPR2CON<5,4>* *Tosc*TMR2 Prescale
t_{on} = 0*250ns*16
t_{on} = 0ms
PWM -> Duty Cycle = 0% -> OFF
ESCREVE "OFF" NO LCD
1


TRATA BOTÃO 1
AÇÃO JÁ FOI EXEC. ?
Sim
Não
FILTRO TERMINOU ?
Não
Sim
CCPR2L = 0x80
LIMPA BITS 4 E 5 DE CCP2CON
t_{on} = CCPR2L:CCPR2CON<5,4>* *Tosc*TMR2 Prescale
t_{on} = 512*250ns*16
t_{on} = 2,048ms
PWM -> Duty Cycle = 50%
ESCREVE "50%" NO LCD
1


TRATA BOTÃO 2
AÇÃO JÁ FOI EXEC. ?
Sim
Não
FILTRO TERMINOU ?
Não
Sim
CCPR2L = 0XC0
LIMPA BITS 4 E 5 DE CCP2CON
t_{on} = CCPR2L:CCPR2CON<5,4>* *Tosc*TMR2 Prescale
t_{on} = 768*250ns*16
t_{on} = 3,072ms
PWM -> Duty Cycle = 75%
ESCREVE "75%" NO LCD
1


TRATA BOTÃO 3
AÇÃO JÁ FOI EXEC. ?
Sim
Não
FILTRO TERMINOU ?
Não
Sim
CCPR2L = 0xFF
SETA BITS 4 E 5 DE CCP2CON
t_{on} = CCPR2L:CCPR2CON<5,4>* *Tosc*TMR2 Prescale
t_{on} = 1023*250ns*16
t_{on} = 4,092ms
PWM -> Duty Cycle = 99,90%
ESCREVE "100%" NO LCD
1

16.5. Código

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Exemplo de Código para utilização com a McMaster            *
 *                              Exemplo 14                                 *
 *                                                                         *
 *                               MOSAICO                                   *
 *                                                                         *
 *  TEL: (0XX11) 4992-8775              SITE: www.mosaico.com.br           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                          *
 *   DATA : 18/01/2010                                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Descrição geral                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

//ESTE EXEMPLO FOI ELABORADO PARA EXPLICAR O FUNCIONAMENTO DO MÓDULO PWM
//DO PIC18F4550. ELE MONITORA OS QUATRO BOTÕES E CONFORME O BOTÃO SELECIONADO
//APLICA UM VALOR DIFERENTE NO PWM, FAZENDO ASSIM UM CONTROLE SOBRE A
//VELOCIDADE DO VENTILADOR. NO LCD É MOSTRADO O VALOR ATUAL DO DUTY CYCLE.
/*
 CONFIGURAÇÂO DOS JUMPERS DA PLACA:

        CAP USB(RC3)                -> ON
        DATA I2C                    -> OFF
        CLOCK I2C                   -> OFF
        COLUNA 2(RB1)               -> ON
        COLUNA 1(RB0)               -> ON
        LINHA 1 / DISPLAY MILHAR(RB4)    -> ON
        LEDS ESPECIAS (RC0,RC1,RC2)      -> ON

*/
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        DEFINIÇÃO PIC                                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <p18F4550.h>           // Register definitions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                INCLUDES DAS FUNÇÕES DE PERIFÉRICOS DO PIC               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <pwm.h>                //PWM library functions
#include <adc.h>                //ADC library functions
#include <timers.h>             //Timer library functions
#include <delays.h>             //Delay library functions
#include <i2c.h>                //I2C library functions
#include <stdlib.h>             //Library functions
#include <usart.h>              //USART library functions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      Configurações para gravação                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#pragma config FOSC = HS
#pragma config CPUDIV = OSC1_PLL2
#pragma config WDT = OFF
#pragma config WDTPS = 128
```

```
#pragma config LVP = OFF
#pragma config PWRT = ON
#pragma config BOR = OFF
#pragma config BORV = 0
#pragma config PBADEN       = OFF
#pragma config DEBUG        = OFF
#pragma config PLLDIV       = 1
#pragma config USBDIV       = 1
#pragma config FCMEN        = OFF
#pragma config IESO  = OFF
#pragma config VREGEN       = OFF
#pragma config MCLRE        = ON
#pragma config LPT1OSC      = OFF
#pragma config CCP2MX       = ON
#pragma config STVREN       = OFF
#pragma config ICPRT        = OFF

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Definição e inicialização das variáveis Globais          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.
unsigned char    FILTRO11;        //FILTRAGEM  1 PARA O BOTÃO 1
unsigned char    FILTRO12;        //FILTRAGEM  2 PARA O BOTÃO 1
unsigned char    FILTRO21;        //FILTRAGEM  1 PARA O BOTÃO 2
unsigned char    FILTRO22;        //FILTRAGEM  2 PARA O BOTÃO 2
unsigned char    FILTRO31;        //FILTRAGEM  1 PARA O BOTÃO 3
unsigned char    FILTRO32;        //FILTRAGEM  2 PARA O BOTÃO 3
unsigned char    FILTRO41;        //FILTRAGEM  1 PARA O BOTÃO 4
unsigned char    FILTRO42;        //FILTRAGEM  2 PARA O BOTÃO 4

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Constantes internas                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de constantes facilita a programação e a manutenção.
#define    T_FILTRO     25

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Declaração dos flags de software                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de flags ajuda na programação e economiza memória RAM.
struct
{
    unsigned BIT0:1;
    unsigned BIT1:1;
    unsigned BIT2:1;
    unsigned BIT3:1;
    unsigned BIT4:1;
    unsigned BIT5:1;
    unsigned BIT7:1;
    unsigned BIT6:1;
}FLAGSbits;            //ARMAZENA OS FLAGS DE CONTROLE

#define    ST_BT1        FLAGSbits.BIT0        //STATUS DO BOTÃO 1
#define    ST_BT2        FLAGSbits.BIT1        //STATUS DO BOTÃO 2
#define    ST_BT3        FLAGSbits.BIT2        //STATUS DO BOTÃO 3
#define    ST_BT4        FLAGSbits.BIT3        //STATUS DO BOTÃO 4

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Definição e inicialização dos port's                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      PROTOTIPAGEM DE FUNÇÕES                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void comando_lcd(unsigned char caracter);
void escreve_lcd(unsigned char caracter);
```

```
void limpa_lcd(void);
void inicializa_lcd(void);
void tela_principal(void);

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              ENTRADAS                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//As entradas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define     BT1     PORTBbits.RB0   //PORTA DO BOTÃO
                                    //1 -> PRESSIONADO
                                    //0 -> LIBERADO

#define     BT2     PORTBbits.RB1   //PORTA DO BOTÃO
                                    //1 -> PRESSIONADO
                                    //0 -> LIBERADO

#define     BT3     PORTBbits.RB2   //PORTA DO BOTÃO
                                    //1 -> PRESSIONADO
                                    //0 -> LIBERADO

#define     BT4     PORTBbits.RB3   //PORTA DO BOTÃO
                                    //1 -> PRESSIONADO
                                    //0 -> LIBERADO

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              SAÍDAS                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//As saídas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define     LINHA_4     PORTBbits.RB7   //PINO PARA ATIVAR LINHA 4 DO TECLADO
                                        //MATRICIAL
                                        //1 -> LINHA ATIVADA
                                        //0 -> LINHA DESATIVADA

#define     rs          PORTEbits.RE0   // via do lcd que sinaliza recepção de dados ou
comando
#define     enable      PORTEbits.RE1   // enable do lcd
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Rotina que envia um COMANDO para o LCD                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void comando_lcd(unsigned char caracter)
{
    rs = 0;                     // seleciona o envio de um comando
    PORTD = caracter;           // carrega o PORTD com o caracter
    enable = 1 ;                // gera pulso no enable
    Delay10TCYx(1);             // espera 3 microsegundos
    enable = 0;                 // desce o pino de enable
    Delay10TCYx(4);             // espera mínimo 40 microsegundos
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               Rotina que envia um DADO a ser escrito no LCD             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void escreve_lcd(unsigned char caracter)
{
    rs = 1;                     // seleciona o envio de um comando
    PORTD = caracter;           // carrega o PORTD com o caracter
    enable = 1;                 // gera pulso no enable
    Delay10TCYx(1);             // espera 10 microsegundos
    enable = 0;                 // desce o pino de enable
    Delay10TCYx(4);             // espera mínimo 40 microsegundos
```

```
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função para limpar o LCD                         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void limpa_lcd(void)
{
    comando_lcd(0x01);          // limpa lcd
    Delay1KTCYx(2);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Inicialização do Display de LCD                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void inicializa_lcd(void)
{
    comando_lcd(0x30);          // envia comando para inicializar display
    Delay1KTCYx(4);             // espera 4 milisengundos

    comando_lcd(0x30);          // envia comando para inicializar display
    Delay10TCYx(10);            // espera 100 microsengundos

    comando_lcd(0x30);          // envia comando para inicializar display

    comando_lcd(0x38);          // liga o display, sem cursor e sem blink

    limpa_lcd();                // limpa lcd

    comando_lcd(0x0c);          // display sem cursor

    comando_lcd(0x06);          // desloca cursor para a direita
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            Tela Principal                                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void tela_principal(void)
{
    limpa_lcd();            // limpa lcd
    comando_lcd(0x83);      // posiciona o cursor na linha 0, coluna 0
    escreve_lcd ('M');      // imprime mensagem no lcd
    escreve_lcd ('C');
    escreve_lcd ('M');
    escreve_lcd ('A');
    escreve_lcd ('S');
    escreve_lcd ('T');
    escreve_lcd ('E');
    escreve_lcd ('R');

    comando_lcd(0xC4);      // posiciona o cursor na linha 1, coluna 4
    escreve_lcd ('P');      // imprime mensagem no lcd
    escreve_lcd ('W');
    escreve_lcd ('M');
    escreve_lcd (':');
    escreve_lcd (' ');
    escreve_lcd ('O');
    escreve_lcd ('F');
    escreve_lcd ('F');
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Tela Tecla 1                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
```

```
void tela_tecla_1(void)
{
    comando_lcd(0xC9);      // posiciona o cursor na linha 0, coluna 0
    escreve_lcd ('O');      // imprime mensagem no lcd
    escreve_lcd ('F');
    escreve_lcd ('F');
    escreve_lcd (' ');
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Tela Tecla 2                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void tela_tecla_2(void)
{
    comando_lcd(0xC9);      // posiciona o cursor na linha 0, coluna 0
    escreve_lcd ('5');      // imprime mensagem no lcd
    escreve_lcd ('0');
    escreve_lcd ('%');
    escreve_lcd (' ');
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Tela Tecla 3                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void tela_tecla_3(void)
{
    comando_lcd(0xC9);      // posiciona o cursor na linha 0, coluna 0
    escreve_lcd ('7');      // imprime mensagem no lcd
    escreve_lcd ('5');
    escreve_lcd ('%');
    escreve_lcd (' ');
}


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Tela Tecla 4                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void tela_tecla_4(void)
{
    comando_lcd(0xC9);      // posiciona o cursor na linha 0, coluna 0
    escreve_lcd ('1');      //   imprime mensagem no lcd
    escreve_lcd ('0');
    escreve_lcd ('0');
    escreve_lcd ('%');
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função Principal                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void main ()
{
    PORTA = 0x00;               //Limpa PORTA
    PORTB = 0x00;               //Limpa PORTB
    PORTC = 0x00;               //Limpa PORTC
    PORTD = 0x00;               //Limpa PORTD
    PORTE = 0x00;               //Limpa PORTE

    LATA = 0x00;                //Limpa PORTA
    LATB = 0x00;                //Limpa PORTB
    LATC = 0x00;                //Limpa PORTC
    LATD = 0x00;                //Limpa PORTD
```

```
    LATE = 0x00;                //Limpa PORTE

    TRISA = 0b00101111;         //CONFIG DIREÇÃO DOS PINOS PORTA0
    TRISB = 0b00001111;         //CONFIG DIREÇÃO DOS PINOS PORTB
    TRISC = 0b10011001;         //CONFIG DIREÇÃO DOS PINOS PORTC
    TRISD = 0b00000000;         //CONFIG DIREÇÃO DOS PINOS PORTD
    TRISE = 0b00000000;         //CONFIG DIREÇÃO DOS PINOS PORTE

    ADCON1 = 0b00001111;        //DESLIGA CONVERSORES A/D


 T2CON = 0x06;

    OpenPWM2(255);


    SetDCPWM2(0);

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      Inicialização do Sistema                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

    RCON = 0X7F;

    LINHA_4 = 1;                //ATIVA LINHA 4 DO TECLADO MATRICIAL

    inicializa_lcd();           // configura o lcd
 tela_principal();


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Rotina Principal                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
    while(1)
    {
        ClrWdt();

        FILTRO11 = 0;           //BT1
        FILTRO12 = T_FILTRO;    //BT1

        FILTRO21 = 0;           //BT2
        FILTRO22 = T_FILTRO;    //BT2

        FILTRO31 = 0;           //BT3
        FILTRO32 = T_FILTRO;    //BT3

        FILTRO41 = 0;           //BT4
        FILTRO42 = T_FILTRO;    //BT4

        while(BT1)
        {
            ClrWdt();
            if (--FILTRO11 == 0)
            {
                FILTRO12--;
                if (FILTRO12 == 0)
                {
                    if (!ST_BT1)
                    {
                        tela_tecla_1();
                        SetDCPWM2(0);
                        ST_BT1 = 1;
                    }
                }
            }
        }
```

```
        ST_BT1 = 0; // BOTÃO LIBERADO, LIMPA O FLAG

        while(BT2)
        {
            ClrWdt();
            if (--FILTRO21 == 0)
            {
                FILTRO22--;
                if (FILTRO22 == 0)
                {
                    if (!ST_BT2)
                    {
                        tela_tecla_2();
                        SetDCPWM2(512);
                        ST_BT2 = 1;
                    }
                }
            }
        }
        ST_BT2 = 0; // BOTÃO LIBERADO, LIMPA O FLAG


        while(BT3)
        {
            ClrWdt();
            if (--FILTRO31 == 0)
            {
                FILTRO32--;
                if (FILTRO32 == 0)
                {
                    if (!ST_BT3)
                    {
                        tela_tecla_3();
                        SetDCPWM2(767);
                        ST_BT3 = 1;
                    }
                }
            }
        }
        ST_BT3 = 0; // BOTÃO LIBERADO, LIMPA O FLAG


        while(BT4)
        {
            ClrWdt();
            if (--FILTRO41 == 0)
            {
                FILTRO42--;
                if (FILTRO42 == 0)
                {
                    if (!ST_BT4)
                    {
                        tela_tecla_4();
                        SetDCPWM2(1023);
                        ST_BT4 = 1;
                    }
                }
            }
        }
        ST_BT4 = 0; // BOTÃO LIBERADO, LIMPA O FLAG

    }// FIM LAÇO PRINCIPAL
}
```

16.6. Dicas e Comentários

Para calcular o valor que deve ser carregado nos registradores que controlam o tempo do pulso em nível alto (duty cycle) a partir de um determinado valor de duty cycle expresso em porcentagem pode-se utilizar a formula a seguir:

$$\textbf{CCPRx} = [(\textbf{PR2})+1] \times 4 \times \textit{Porcentagem desejada}$$

Vejamos como exemplo o valor de 50% adotado nesta experiência. A porcentagem desejada é de 50% e o valor de **PR2** é 255, assim,

$$\textbf{CCPRx} = 256 * 4 * 0,5$$

$$\textbf{CCPRx} = 512$$

Ou seja, se carregarmos os registradores que controlam o tempo do pulso em nível alto com 512, obteremos um duty cycle de 50%

16.7. Exercícios Propostos

1. Corrija o problema encontrado no nível 100%, evitando que a saída seja colocada em zero, mesmo que por um período de tempo muito curto.
2. Em vez de trabalhar com somente 4 níveis de PWM, altere o sistema para que um botão ligue e desligue a saída e outros dois botões incremente e decremente o PWM, de 50 a 100% com passos de 5%.
3. Ative as duas saídas PWMs ao mesmo tempo, uma para o ventilador e outra para a resistência. Utilize dois botões para controlar o ajuste de cada uma delas.

17. Experiência 15 – Acesso às memórias de dados e programa

17.1. Objetivo

O acesso à memória de dados EEPROM já foi visto na experiência 6, portanto, a novidade desta experiência é o acesso à memória de programa do microcontrolador.

17.2. Descrição

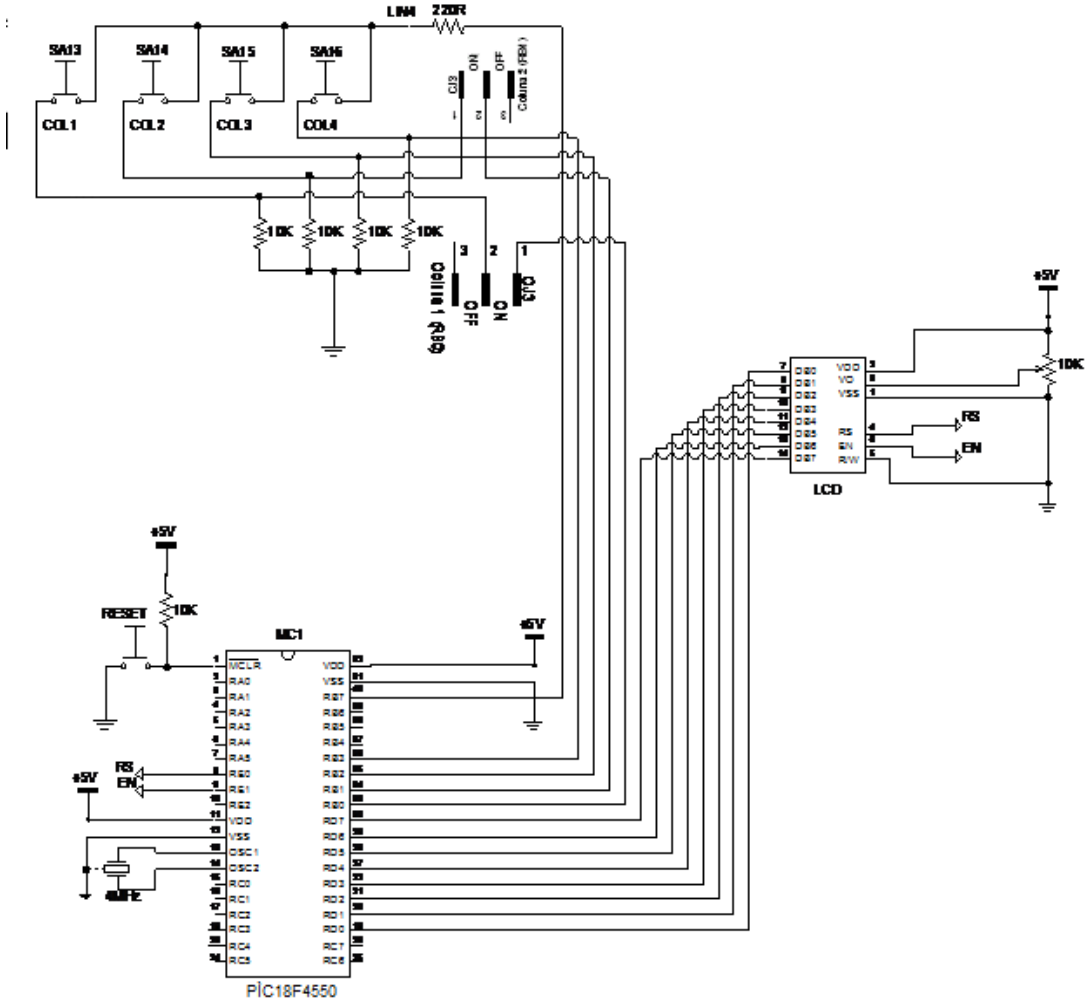
Nesta experiência o LCD está dividido em duas partes. Do lado esquerdo temos um valor relativo à memória de dados, variável de 0 a FFh (8-bits), com incremento e decremento rotativo através dos botões das colunas 2 e 3. Do lado direito o valor será para a memória de programa, também com incremento e decremento rotativo através dos botões das colunas 2 e 3, podendo ir de 0 a 3FFFh (14-bits).

Para alterar o controle dos botões entre o lado esquerdo e o lado direito deve ser usado o botão da coluna 1., sendo que o lado ativo no momento é indicado entre os sinais > e <.

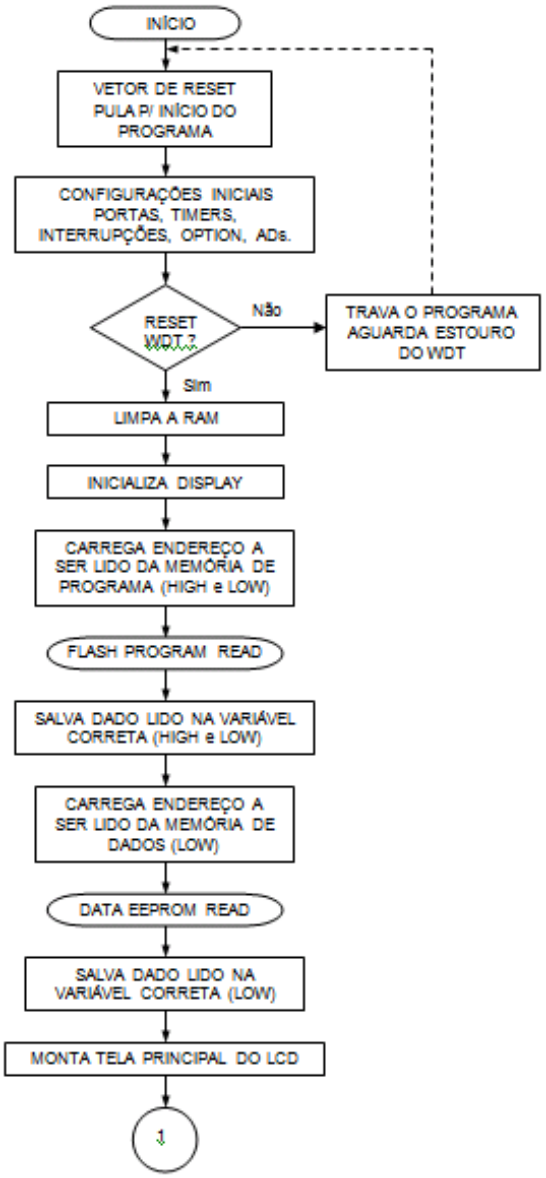
Depois de ajustados os valores desejados, basta pressionar o botão da coluna 4 para que ambos valores sejam gravados, cada um na memória correspondente.

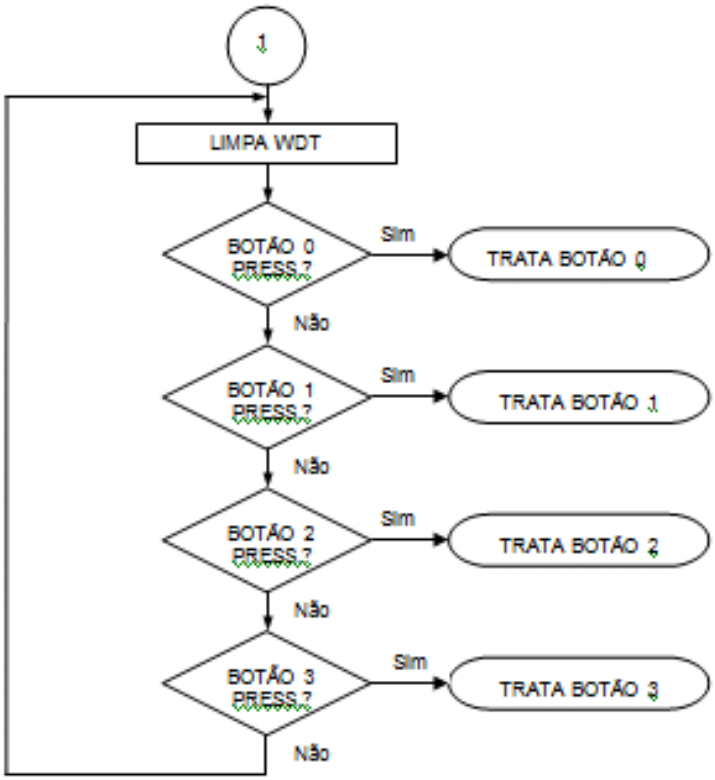
Para checar a gravação, altere os valores e reset o sistema (botão de reset). Os valores gravados serão recuperados na inicialização e mostrados no LCD.

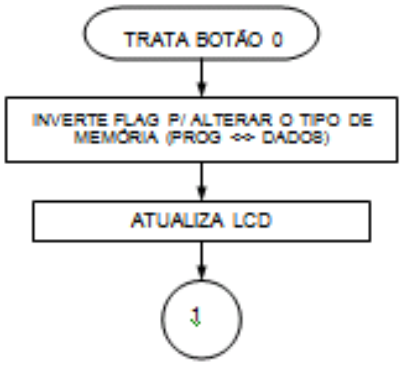
17.3. Esquema Elétrico

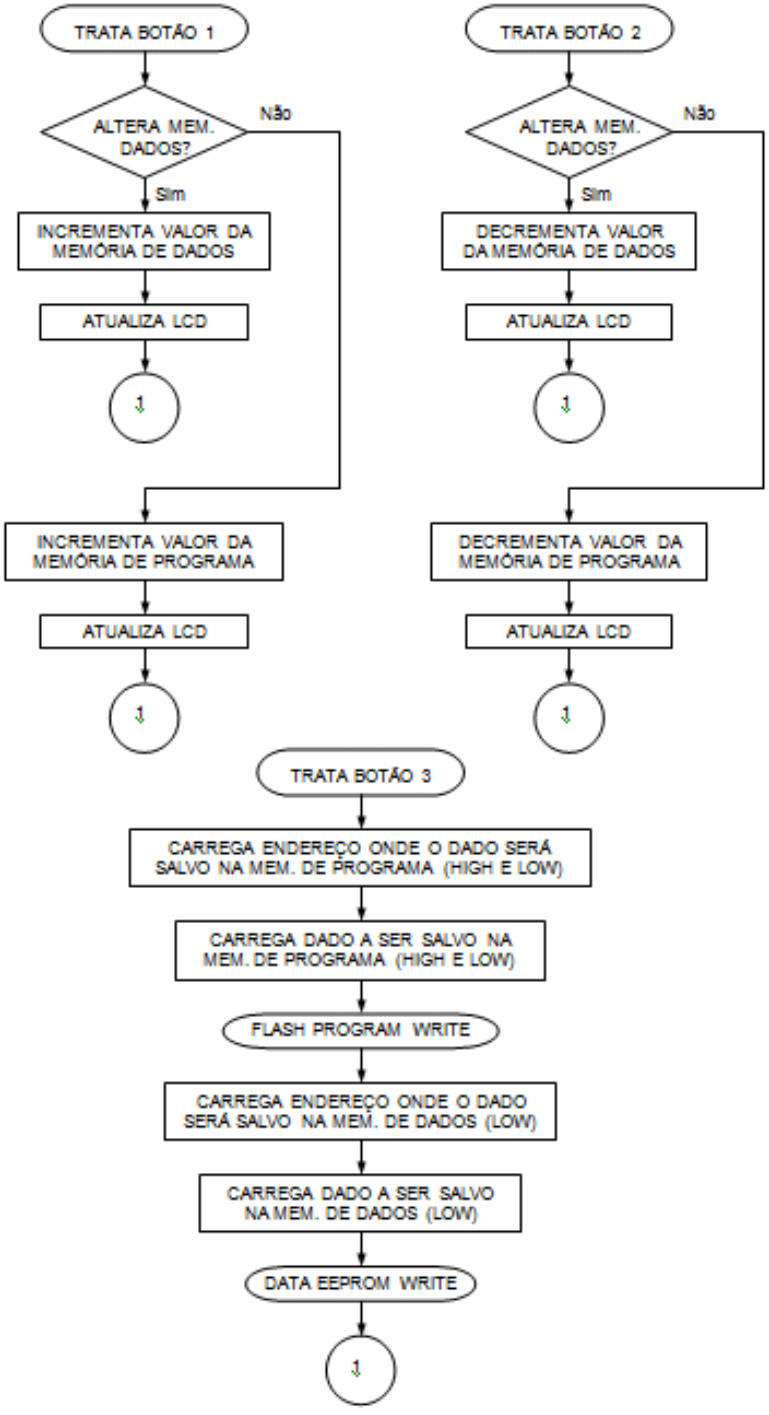


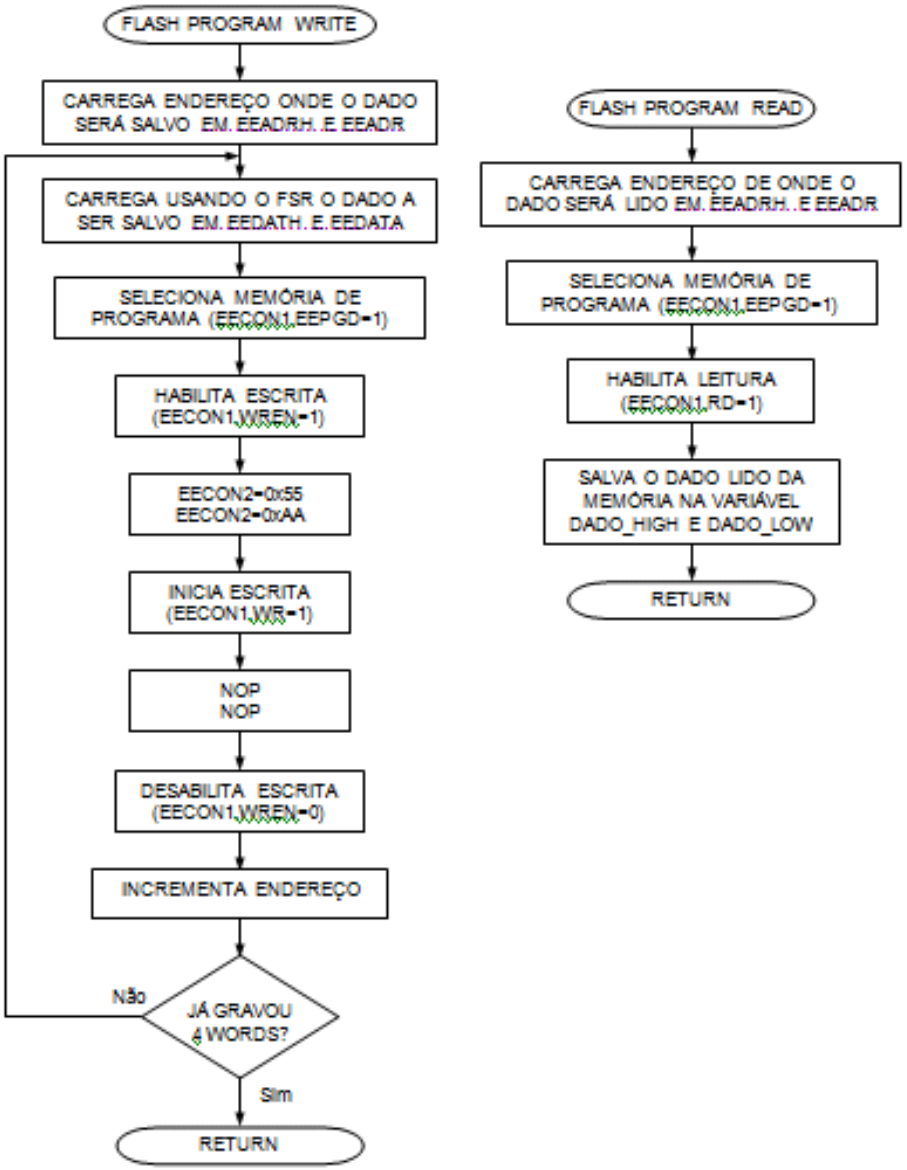
17.4. Fluxograma

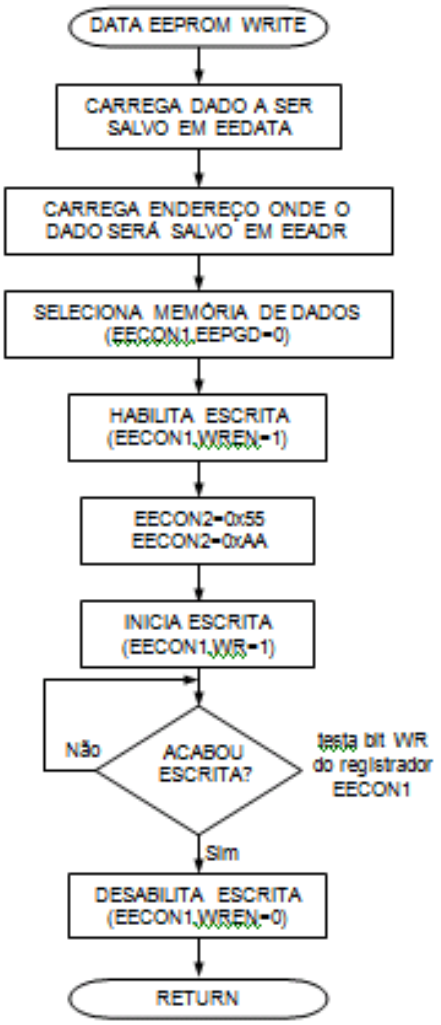

INÍCIO
VETOR DE RESET PULA P/ INÍCIO DO PROGRAMA
CONFIGURAÇÕES INICIAIS PORTAS, TIMERS, INTERRUPÇÕES, OPTION, ADs.
RESET WDT ?
Não
TRAVA O PROGRAMA AGUARDA ESTOURO DO WDT
Sim
LIMPA A RAM
INICIALIZA DISPLAY
CARREGA ENDEREÇO A SER LIDO DA MEMÓRIA DE PROGRAMA (HIGH e LOW)
FLASH PROGRAM READ
SALVA DADO LIDO NA VARIÁVEL CORRETA (HIGH e LOW)
CARREGA ENDEREÇO A SER LIDO DA MEMÓRIA DE DADOS (LOW)
DATA EEPROM READ
SALVA DADO LIDO NA VARIÁVEL CORRETA (LOW)
MONTA TELA PRINCIPAL DO LCD
1

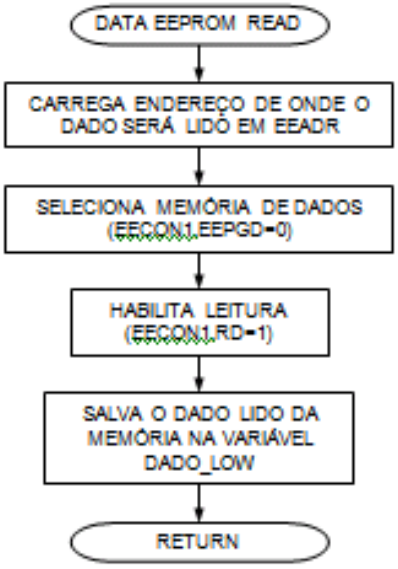

1
LIMPA WDT
BOTÃO 0 PRESS.?
Sim
TRATA BOTÃO 0
Não
BOTÃO 1 PRESS.?
Sim
TRATA BOTÃO 1
Não
BOTÃO 2 PRESS.?
Sim
TRATA BOTÃO 2
Não
BOTÃO 3 PRESS.?
Sim
TRATA BOTÃO 3
Não


TRATA BOTÃO 0
INVERTE FLAG P/ ALTERAR O TIPO DE MEMÓRIA (PROG <-> DADOS)
ATUALIZA LCD
1


TRATA BOTÃO 1
ALTERA MEM. DADOS?
Não
Sim
INCREMENTA VALOR DA MEMÓRIA DE DADOS
ATUALIZA LCD
1
INCREMENTA VALOR DA MEMÓRIA DE PROGRAMA
ATUALIZA LCD
1
TRATA BOTÃO 2
ALTERA MEM. DADOS?
Não
Sim
DECREMENTA VALOR DA MEMÓRIA DE DADOS
ATUALIZA LCD
1
DECREMENTA VALOR DA MEMÓRIA DE PROGRAMA
ATUALIZA LCD
1
TRATA BOTÃO 3
CARREGA ENDEREÇO ONDE O DADO SERÁ SALVO NA MEM. DE PROGRAMA (HIGH E LOW)
CARREGA DADO A SER SALVO NA MEM. DE PROGRAMA (HIGH E LOW)
FLASH PROGRAM WRITE
CARREGA ENDEREÇO ONDE O DADO SERÁ SALVO NA MEM. DE DADOS (LOW)
CARREGA DADO A SER SALVO NA MEM. DE DADOS (LOW)
DATA EEPROM WRITE
1


FLASH PROGRAM WRITE
CARREGA ENDEREÇO ONDE O DADO SERÁ SALVO EM EEADRH. E EEADR
CARREGA USANDO O FSR O DADO A SER SALVO EM EEDATH. E EEDATA
SELECIONA MEMÓRIA DE PROGRAMA (EECON1.EEPGD=1)
HABILITA ESCRITA (EECON1.WREN=1)
EECON2=0x55
EECON2=0xAA
INICIA ESCRITA (EECON1.WR=1)
NOP
NOP
DESABILITA ESCRITA (EECON1.WREN=0)
INCREMENTA ENDEREÇO
Não
JÁ GRAVOU 4 WORDS?
Sim
RETURN
FLASH PROGRAM READ
CARREGA ENDEREÇO DE ONDE O DADO SERÁ LIDO EM EEADRH. E EEADR
SELECIONA MEMÓRIA DE PROGRAMA (EECON1.EEPGD=1)
HABILITA LEITURA (EECON1.RD=1)
SALVA O DADO LIDO DA MEMÓRIA NA VARIÁVEL DADO_HIGH E DADO_LOW
RETURN


DATA EEPROM WRITE
CARREGA DADO A SER SALVO EM EEDATA
CARREGA ENDEREÇO ONDE O DADO SERÁ SALVO EM EEADR
SELECIONA MEMÓRIA DE DADOS (EECON1.EEPGD=0)
HABILITA ESCRITA (EECON1.WREN=1)
EECON2=0x55
EECON2=0xAA
INICIA ESCRITA (EECON1.WR=1)
Não
ACABOU ESCRITA?
Testa bit WR do registrador EECON1
Sim
DESABILITA ESCRITA (EECON1.WREN=0)
RETURN


DATA EEPROM READ
CARREGA ENDEREÇO DE ONDE O DADO SERÁ LIDO EM EEADR
SELECIONA MEMÓRIA DE DADOS (EECON1.EEPGD=0)
HABILITA LEITURA (EECON1.RD=1)
SALVA O DADO LIDO DA MEMÓRIA NA VARIÁVEL DADO_LOW
RETURN

17.5. Código

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Exemplo de Código para utilização com a McMaster             *
 *                                 Exemplo 15                                *
 *                                                                           *
 *                                  MOSAICO                                  *
 *                                                                           *
 *  TEL: (0XX11) 4992-8775                 SITE: www.mosaico.com.br          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                            *
 *   DATA : 18/01/2010                                                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              Descrição geral                              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*
ESTE EXEMPLO FOI ELABORADO PARA EXPLICAR O FUNCIONAMENTO DA LEITURA/ESCRITA
TANTO NA MEMÓRIA DE DADOS QUANTO NA MEMÓRIA DE PROGRAMA.
OS BOTÕES PARA A UTILIZAÇÃO DO EXEMPLO ESTÃO NA LINHA 4 COMO SEGUE:

 COLUNA1      ->     ALTERNA O TIPO DE MEMÓRIA
 COLUNA2 ->    INCREMENTA O VALOR QUE SERÀ SALVO NA MEMÓRIA
 COLUNA3 ->    DECREMENTA O VALOR QUE SERÀ SALVO NA MEMÓRIA
 COLUNA4 ->  SALVA O VALOR INDICADO NA MEMÓRIA SELECIONADA

REINICIE O EQUIPAMENTO E OBSERVANDO O RESULTADO NO LCD.
*/
/*
 CONFIGURAÇÃO DOS JUMPERS DA PLACA:

        CAP USB(RC3)              -> ON
        DATA I2C                  -> OFF
        CLOCK I2C                 -> OFF
        COLUNA 2(RB1)             -> ON
        COLUNA 1(RB0)             -> ON
        LINHA 1 / DISPLAY MILHAR(RB4)    -> ON
        LEDS ESPECIAS (RC0,RC1,RC2)      -> ON

*/


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        DEFINIÇÃO PIC                      *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <p18F4550.h>            // Register definitions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               INCLUDES DAS FUNÇÕES DE PERIFÉRICOS DO PIC              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <pwm.h>                //PWM library functions
#include <adc.h>                //ADC library functions
#include <timers.h>             //Timer library functions
#include <delays.h>             //Delay library functions
#include <i2c.h>                //I2C library functions
#include <stdlib.h>             //Library functions
#include <usart.h>              //USART library functions
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      Configurações para gravação                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#pragma config FOSC = HS
#pragma config CPUDIV = OSC1_PLL2
#pragma config WDT = OFF
#pragma config WDTPS = 128
#pragma config LVP = OFF
#pragma config PWRT = ON
#pragma config BOR = OFF
#pragma config BORV = 0
#pragma config PBADEN = OFF
#pragma config DEBUG = OFF
#pragma config PLLDIV = 1
#pragma config USBDIV = 1
#pragma config FCMEN = OFF
#pragma config IESO = OFF
#pragma config VREGEN = OFF
#pragma config MCLRE = ON
#pragma config LPT1OSC = OFF
#pragma config CCP2MX = ON
#pragma config STVREN = OFF
#pragma config ICPRT = OFF

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Definição e inicialização das variáveis Globais          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.

unsigned char    FILTRO;        //FILTRO BOTÃO

char mem[8];

unsigned char    tempo_turbo = 30;
unsigned char    cont_mem_dados;
unsigned short   cont_mem_prog;
unsigned char    x;

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Constantes internas                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de constantes facilita a programação e a manutenção.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Declaração dos flags de software                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de flags ajuda na programação e economiza memória RAM.
struct
{
    unsigned BIT0:1;
    unsigned BIT1:1;
    unsigned BIT2:1;
    unsigned BIT3:1;
    unsigned BIT4:1;
    unsigned BIT5:1;
    unsigned BIT6:1;
    unsigned BIT7:1;
}FLAGSbits;            //ARMAZENA OS FLAGS DE CONTROLE

#define    TIPO_MEM    FLAGSbits.BIT0

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Definição e inicialização dos port's               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      PROTOTIPAGEM DE FUNÇÕES                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void comando_lcd(unsigned char caracter);
void escreve_lcd(unsigned char caracter);
void limpa_lcd(void);
void inicializa_lcd(void);
void escreve_frase(const rom char *frase);
void atualiza_linha_1(void);
void atualiza_linha_2(void);
void write_EEPROM(unsigned char endereco, unsigned char dado);
unsigned char read_EEPROM(unsigned char endereco);
void write_FLASH(unsigned addr, char *buffer);
unsigned short read_FLASH(unsigned endereco);

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                               ENTRADAS                                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// As entradas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define BT1 PORTBbits.RB0       //BOTÃO 1
                                //0 -> PRESSIONADO
                                //1 -> LIBERADO

#define BT2 PORTBbits.RB1       //BOTÃO 2
                                //0 -> PRESSIONADO
                                //1 -> LIBERADO

#define BT3 PORTBbits.RB2       //BOTÃO 3
                                //0 -> PRESSIONADO
                                //1 -> LIBERADO

#define BT4 PORTBbits.RB3       //BOTÃO 4
                                //0 -> PRESSIONADO
                                //1 -> LIBERADO

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                               SAÍDAS                                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//As saídas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define     LINHA_4     PORTBbits.RB7   //PINO PARA ATIVAR LINHA 4 DO TECLADO
                                        //MATRICIAL
                                        //1 -> LINHA ATIVADA
                                        //0 -> LINHA DESATIVADA


#define rs        PORTEbits.RE0    // via do lcd que sinaliza recepção de dados ou
comando
#define enable    PORTEbits.RE1    // enable do lcd

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               Rotina que envia um COMANDO para o LCD                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void comando_lcd(unsigned char caracter)
{
    rs = 0;                         // seleciona o envio de um comando
    PORTD = caracter;               // carrega o PORTD com o caracter
    enable = 1 ;                    // gera pulso no enable
    Delay10TCYx(1);                 // espera 10 microsegundos
    enable = 0;                     // desce o pino de enable
```

```
    Delay10TCYx(4);                 // espera mínimo 40 microsegundos
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Rotina que envia um DADO a ser escrito no LCD               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void escreve_lcd(unsigned char caracter)
{
    rs = 1;                     // seleciona o envio de um comando
    PORTD = caracter;           // carrega o PORTD com o caracter
    enable = 1;                 // gera pulso no enable
    Delay10TCYx(1);             // espera 10 microsegundos
    enable = 0;                 // desce o pino de enable
    Delay10TCYx(4);             // espera mínimo 40 microsegundos
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                             Função para limpar o LCD                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void limpa_lcd(void)
{
    comando_lcd(0x01);              // limpa lcd
    Delay1KTCYx(2);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Inicialização do Display de LCD                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void inicializa_lcd(void)
{
    comando_lcd(0x30);              // envia comando para inicializar display
    Delay1KTCYx(4);                 // espera 4 milisengundos

    comando_lcd(0x30);              // envia comando para inicializar display
    Delay10TCYx(10);                // espera 100 microsengundos

    comando_lcd(0x30);              // envia comando para inicializar display

    comando_lcd(0x38);              // liga o display, sem cursor e sem blink

    limpa_lcd();                    // limpa lcd

    comando_lcd(0x0c);              // display sem cursor

    comando_lcd(0x06);              // desloca cursor para a direita
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Função para escrever uma frase no LCD                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void escreve_frase(const rom char *frase)
{
    do
    {
        escreve_lcd(*frase);
    }while(*++frase);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                FUNÇÃO PARA ATUALIZAR A LINHA 1 DO LCD                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
```

```
void atualiza_linha_1(void)
{
    limpa_lcd();                // limpa lcd
    comando_lcd(0x80);          // posiciona o cursor na linha 0, coluna 0
                                // imprime mensagem no lcd
    escreve_frase("M.DADOS  M.PROG.");
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               FUNÇÃO PARA ATUALIZAR A LINHA 2 DO LCD                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void atualiza_linha_2(void)
{
    union
    {
        struct
        {
            unsigned short low:8;
            unsigned short high:8;
        } ;
        struct
        {
            unsigned short all;
        } ;
    }dado;

    unsigned char aux;

    comando_lcd(0xC1);                  //POSICIONA CURSOR NA LINHA 1, COLUNA 1

    if (TIPO_MEM) escreve_lcd('>');     //SE TIPO DE MEMÓRIA = 1, IMPRIME '>'
    else escreve_lcd(' ');              //SE TIPO DE MEMÓRIA = 0, IMPRIME ' '

    aux = cont_mem_dados >> 4;          //CONVERTE cont_mem_dados EM HEXADECIMAL
    if(aux < 10) aux = aux + 0x30;
    else aux = aux + 0x37;
    escreve_lcd(aux);

    aux = cont_mem_dados & 0b00001111;
    if(aux < 10) aux = aux + 0x30;
    else aux = aux + 0x37;
    escreve_lcd(aux);

    if (TIPO_MEM) escreve_lcd('<');     //SE TIPO DE MEMÓRIA = 1, IMPRIME '<'
    else escreve_lcd(' ');              //SE TIPO DE MEMÓRIA = 0, IMPRIME ' '

    dado.all = cont_mem_prog;

    escreve_lcd(' ');
    escreve_lcd(' ');
    escreve_lcd(' ');

    if (!TIPO_MEM) escreve_lcd('>');        //SE TIPO DE MEMÓRIA = 0, IMPRIME '>'
    else escreve_lcd(' ');                  //SE TIPO DE MEMÓRIA = 1, IMPRIME ' '

    aux = dado.high >> 4;                   //CONVERTE cont_mem_prog EM HEXADECIMAL
    if(aux < 10) aux = aux + 0x30;
    else aux = aux + 0x37;
    escreve_lcd(aux);

    aux = dado.high & 0b00001111;
    if(aux < 10) aux = aux + 0x30;
    else aux = aux + 0x37;
    escreve_lcd(aux);
```

```
    aux = dado.low >> 4;
    if(aux < 10) aux = aux + 0x30;
    else aux = aux + 0x37;
    escreve_lcd(aux);

    aux = dado.low & 0b00001111;
    if(aux < 10) aux = aux + 0x30;
    else aux = aux + 0x37;
    escreve_lcd(aux);

    if (!TIPO_MEM) escreve_lcd('<');      //SE TIPO DE MEMÓRIA = 0, IMPRIME '<'
    else escreve_lcd(' ');                //SE TIPO DE MEMÓRIA = 1, IMPRIME ' '
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Função de escrita da EEPROM interna                      *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void write_EEPROM(unsigned char endereco, unsigned char dado)
{
    EEDATA = dado;                //carrega dado
    EEADR = endereco;             //carrega endereço

    EECON1bits.CFGS = 0;          //habilita acesso a EEPROM
    EECON1bits.EEPGD = 0;         //aponta para memória de dados
    EECON1bits.WREN = 1;          //habilita escrita
    INTCONbits.GIE = 0;           //desabilita todas as interrupções
    EECON2 = 0x55;                //escreve 0x55 em EECON2 (obrigatório)
    EECON2 = 0xAA;                //escreve 0xAA em EECON2 (obrigatório)
    EECON1bits.WR = 1;            //inicia a escrita
    INTCONbits.GIE = 1;           //habilita todas as interrupções
    while(EECON1bits.WR);         //aguarda bit WR de EECON1 = 0
    EECON1bits.WREN = 0;          //desabilita escrita
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Função de leitura da EEPROM interna                      *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

unsigned char read_EEPROM(unsigned char endereco)
{
    EEADR = endereco;             //carrega endereço

    EECON1bits.CFGS = 0;          //habilita acesso a EEPROM
    EECON1bits.EEPGD = 0;         //aponta para memória de dados
    EECON1bits.RD = 1;            //habilita leitura

    return(EEDATA);               //retorna dado disponível em EEDATA
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Função de escrita da FLASH interna                      *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

// OBS: ESTA FUNÇÃO, O ENDEREÇO DEVE TER FINAL 0 OU 8
// EX.: 0x500, 0x508, 0x60, 0x68 ...

void write_FLASH(unsigned addr, char *buffer)
{
    unsigned char i;

    TBLPTR = addr;                 //carrega endereço

    EECON1bits.EEPGD = 1;          //aponta para memória FLASH
    EECON1bits.CFGS = 0;           //habilita acesso a FLASH
    EECON1bits.WREN = 1;           //habilita escrita
```

```
    EECON1bits.FREE = 1;          //apaga FLASH a partir do endereço apontado no TBLPTR
    INTCONbits.GIE = 0;           //desabilita todas as interrupções
    EECON2 = 0x55;                //escreve 0x55 em EECON2 (obrigatório)
    EECON2 = 0xAA;                //escreve 0xAA em EECON2 (obrigatório)
    EECON1bits.WR = 1;            //inicializa escrita da FLASH
    INTCONbits.GIE = 1;           //habilita todas as interrupções
    while(EECON1bits.WR);
    EECON1bits.WREN = 0;          //desabilita escrita

    for(i = 0;i < 8;i++)
    {
        TABLAT = buffer[i];       //carrega dado em TABLAT

        _asm
        TBLWTPOSTINC              //escreve dado carregado em TABLAT e incrementa
TBLPTR
        _endasm

        if(((i + 1) % 8) == 0)    //identifica os grupos de 8 bytes e habilita a
escrita
        {
            TBLPTR -= 8;          //retorna 8 posições em TBLPTR

            EECON1bits.EEPGD = 1; //aponta para memória FLASH
            EECON1bits.CFGS = 0;  //habilita acesso a FLASH
            EECON1bits.WREN = 1;  //habilita escrita
            EECON1bits.FREE = 0;  //somente escrita da FLASH
            INTCONbits.GIE = 0;   //desabilita todas as interrupções
            EECON2 = 0x55;        //escreve 0x55 em EECON2 (obrigatório)
            EECON2 = 0xAA;        //escreve 0xAA em EECON2 (obrigatório)
            EECON1bits.WR = 1;    //inicializa escrita da FLASH
            INTCONbits.GIE = 1;   //habilita todas as interrupções
            while(EECON1bits.WR);
            EECON1bits.WREN = 0;  //desabilita escrita

            TBLPTR += 8;          //retorna à posição original em TBLPTR
        }
    }
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Função de leitura da FLASH interna                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

unsigned short read_FLASH(unsigned endereco)
{
    union
    {
        struct
        {
            unsigned short low:8;
            unsigned short high:8;
        } ;
        struct
        {
            unsigned short all;
        } ;
    }dado;

    TBLPTR = endereco;        //carrega endereço

    _asm                      //lê posição apontada pelo TBLPTR, envia
    TBLRDPOSTINC              //dado para TABLAT e incremanta TBLPTR
    _endasm

    dado.low = TABLAT;        //salva dado em dado.low
```

```
    _asm                        //lê posição apontada pelo TBLPTR, envia
    TBLRDPOSTINC                //dado para TABLAT e incremanta TBLPTR
    _endasm

    dado.high = TABLAT;         //salva dado em dado.high

    return(dado.all);           //retorna dado disponível em dado.all
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função Principal                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void main ()
{
    PORTA = 0x00;               //Limpa PORTA
    PORTB = 0x00;               //Limpa PORTB
    PORTC = 0x00;               //Limpa PORTC
    PORTD = 0x00;               //Limpa PORTD
    PORTE = 0x00;               //Limpa PORTE

    LATA = 0x00;                //Limpa PORTA
    LATB = 0x00;                //Limpa PORTB
    LATC = 0x00;                //Limpa PORTC
    LATD = 0x00;                //Limpa PORTD
    LATE = 0x00;                //Limpa PORTE

    TRISA = 0b11111111;                 //CONFIG DIREÇÃO DOS PINOS PORTA
    TRISB = 0b00001111;                 //CONFIG DIREÇÃO DOS PINOS PORTB
    TRISC = 0b11111111;                 //CONFIG DIREÇÃO DOS PINOS PORTC
    TRISD = 0b00000000;                 //CONFIG DIREÇÃO DOS PINOS PORTD
    TRISE = 0b00000100;                 //CONFIG DIREÇÃO DOS PINOS PORTE

    ADCON1 = 0b00001111;                //DESLIGA CONVERSORES A/D

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      Inicialização do Sistema                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

    LINHA_4 = 1;                //ATIVA LINHA 4 DO TECLADO MATRICIAL

    inicializa_lcd();            // configura o lcd

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Rotina Principal                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

    cont_mem_dados = read_EEPROM(0x10);     //LÊ MEMÓRIA EEPROM
    cont_mem_prog = read_FLASH(0x800);      //LÊ MEMÓRIA FLASH

    for(x=0;x<8;x++) mem[x]=0;             //INICIA BUFFER PARA GRAVAÇÃO
                                           //DA MEMÓRIA FLASH

    atualiza_linha_1();                     //ATUALIZA LINHA 1
    atualiza_linha_2();                     //ATUALIZA LINHA 2

    while(1)
    {
        ClrWdt();                          //LIMPA O WDT

        if(BT1)                            //SE BOTÃO 1 PRESSIONADO
        {
            if (FILTRO)                    //AÇÃO DO BOTÃO FOI EXECUTADA?
            {
                FILTRO--;                  //DECREMENTA FILTRO
```

```
                if(!FILTRO)              //TERMINOU FILTRO?
                {
                    if(TIPO_MEM) TIPO_MEM = 0;    //SE TIPO DE MEM = 1, TIPO DE MEM = 0
                    else TIPO_MEM = 1;            //SE NÃO, TIPO DE MEM = 1
                    atualiza_linha_2();           //ATUALIZA LINHA 2 DO LCD
                }
            }
            continue;
        }

        if(BT2)                          //SE BOTÃO 2 PRESSIONADO
        {
            FILTRO--;                    //DECREMENTA FILTRO
            if (!FILTRO)                 //TERMINOU FILTRO?
            {
                tempo_turbo--;
                if (!tempo_turbo)
                {
                    if (TIPO_MEM) cont_mem_dados++;    //SE TIPO DE MEM = 1, INCREMENTA
cont_mem_dados
                    else cont_mem_prog++;              //SE NÃO, INCREMENTA
cont_mem_prog
                    atualiza_linha_2();                //ATUALIZA LINHA 2 DO LCD
                    tempo_turbo = 30;                  //REINICIA TURBO DE TECLA
                }
            }
            continue;
        }

        if(BT3)                          //SE BOTÃO 3 PRESSIONADO
        {
            FILTRO--;                    //DECREMENTA FILTRO
            if (!FILTRO)                 //TERMINOU FILTRO?
            {
                tempo_turbo--;
                if (!tempo_turbo)
                {
                    if (TIPO_MEM) cont_mem_dados--;    //SE TIPO DE MEM = 1, DECREMENTA
cont_mem_dados
                    else cont_mem_prog--;              //SE NÃO, DECREMENTA
cont_mem_prog
                    atualiza_linha_2();                //ATUALIZA LINHA 2 DO LCD
                    tempo_turbo = 30;                  //REINICIA TURBO DE TECLA
                }
            }
            continue;
        }


        if(BT4)                          //SE BOTÃO 4 PRESSIONADO
        {
            if (FILTRO)                  //AÇÃO DO BOTÃO FOI EXECUTADA?
            {
                FILTRO--;                //DECREMENTA FILTRO
                if(!FILTRO)              //TERMINOU FILTRO?
                {
                    write_EEPROM(0x10, cont_mem_dados);    //GRAVA NA EEPROM E FLASH
                    mem[0] = cont_mem_prog;
                    mem[1] = cont_mem_prog >> 8;
                    write_FLASH(0x800, &mem);
                }
            }
            continue;
        }

        FILTRO = 255;            //RECARREGA FILTRO DE BOTÕES
```

```
    }// FIM LAÇO PRINCIPAL
}
```

17.6. Dicas e Comentários

Note que a rotina que grava o dado na memória de programa, na realidade, grava o valor de 4 words. Porém, o software da experiência altera o valor de apenas uma dessas words. O problema é que a gravação de informações na memória de programa, no caso do PIC18F4550, sempre deve ser realizada de 4 em 4 words. Desta forma, não é possível gravar apenas um valor ou uma word. Sempre a gravação será de 4 words no mínimo. Portanto, sempre que se desejar gravar menos do que 4 words mantendo as outras intactas. Deve-se inicialmente realizar uma leitura de todas as 4 words, alterar as que se desejarem e regravas todas de uma única vez.

17.7. Exercícios Propostos

1. Crie um novo sistema onde do lado esquerdo você continua informando o dado para a E^2PROM, no centro você escolhe a posição, de 0 a 255 e do lado direito é informada o tipo de operação: “E” para escrita e “L” para leitura. O botão da coluna 1 continua alterando entre os parâmetros a serem ajustados e os botões das colunas 2 e 3 alteram o parâmetro atual. O botão da coluna 4 efetua a operação de escrita ou leitura dependendo da seleção ajustada no LCD.
2. Repita o exercício anterior para a memória de programa FLASH;
3. Crie um programa que copie internamente todo o código de programa para outra posição. Por exemplo, copie todas a página 0 da memória de programa para a página 1. Depois, utilize o gravador e o Mplab para ler a memória de programa e verificar se a operação foi executada com sucesso.

18. Experiência 16 – Master I²C

18.1. Objetivo

O objetivo desta experiência é mostrar ao aluno como acessar a memória de dados EEPROM externa (24LC256) utilizando os recursos de hardware do PIC para implementar o protocolo de comunicação I²C.

18.2. Descrição

Conforme comentado na descrição do hardware, está memória está mapeada no endereço 7h (111b) da rede de comunicação I²C a fim de evitar conflitos com o relógio de tempo real. Além disso, como a memória utilizada no McMaster é de 256Kbits, ou seja, 32Kbytes, são necessários 15 bits, ou seja, 2 bytes, para o correto endereçamento.

Levando-se isso em consideração, para a escrita de dados na memória o microcontrolador deverá enviar a seguinte seqüência de informações:

- Envia o byte de controle que deve incorporar o endereço de hardware da memória na rede I2C além do bit de R/W, que neste caso deve ser enviado em 0 a fim de sinalizar uma operação de escrita. Assim, o byte de controle completo considerando o mapeamento adotado no McMaster é 10101110b;
- Envia um start bit;
- Recebe o bit de acknowledge (ACK);
- Envia a parte mais significativa do endereço onde o dado será gravado;
- Recebe o bit de acknowledge (ACK);
- Envia a parte menos significativa do endereço onde o dado será gravado;
- Recebe o bit de acknowledge (ACK);
- Envia o dado a ser gravado;
- Recebe o bit de acknowledge (ACK);
- Envia um stop bit.

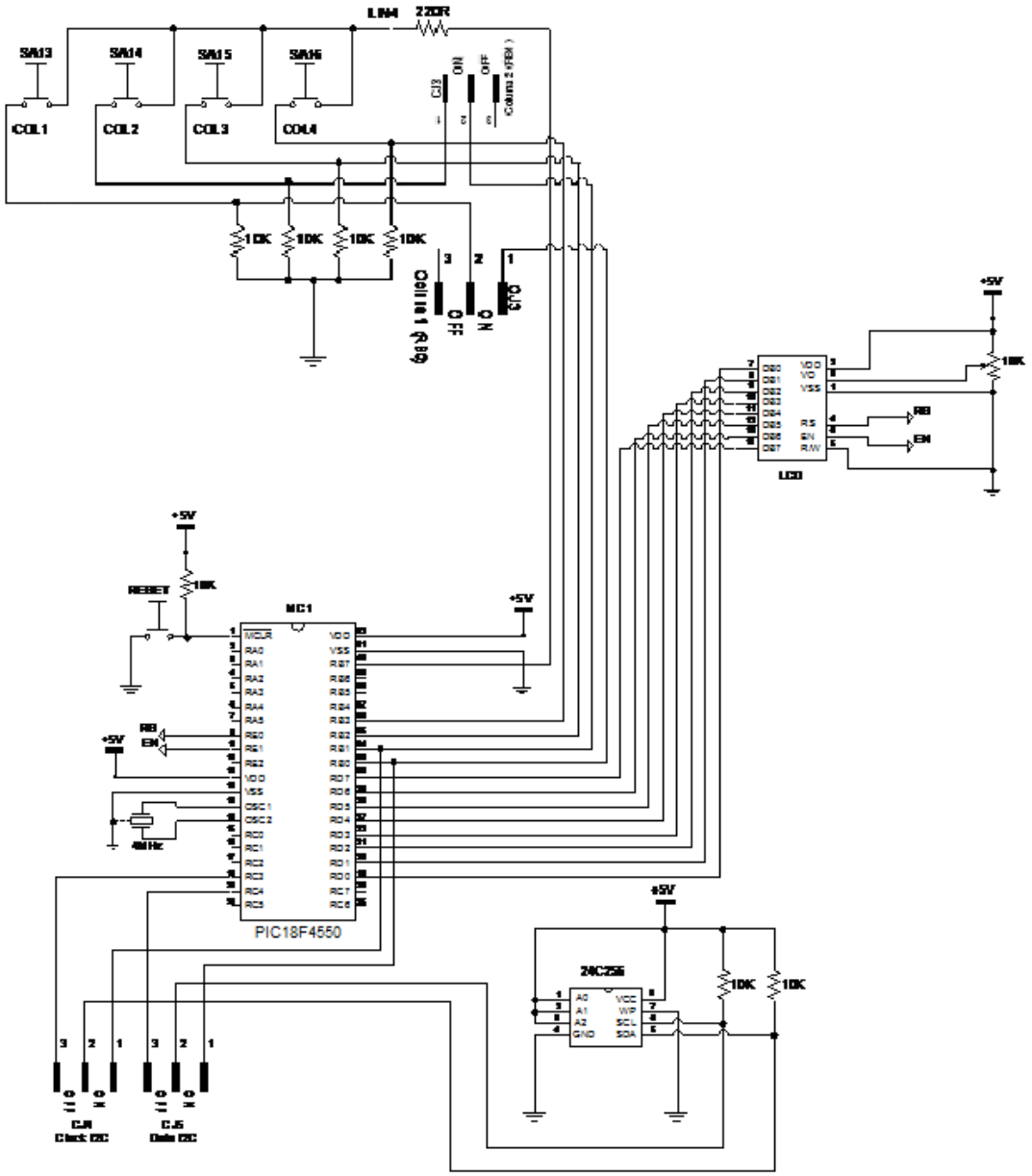
Já para uma operação de leitura a seqüência deverá ser:

- Envia um start bit;
- Envia o byte de controle que deve incorporar o endereço de hardware da memória na rede I2C além do bit de R/W, que neste caso deve ser enviado em 0 a fim de sinalizar uma operação de escrita. Note que inicialmente o endereço que se deseja ler deve ser escrito na memória. Assim, o byte de controle completo considerando o mapeamento adotado no McMaster é 10101110b;
- Recebe o bit de acknowledge (ACK);
- Envia a parte mais significativa do endereço de o dado será lido;
- Recebe o bit de acknowledge (ACK);
- Envia a parte menos significativa do endereço de onde o dado será lido;
- Recebe o bit de acknowledge (ACK);

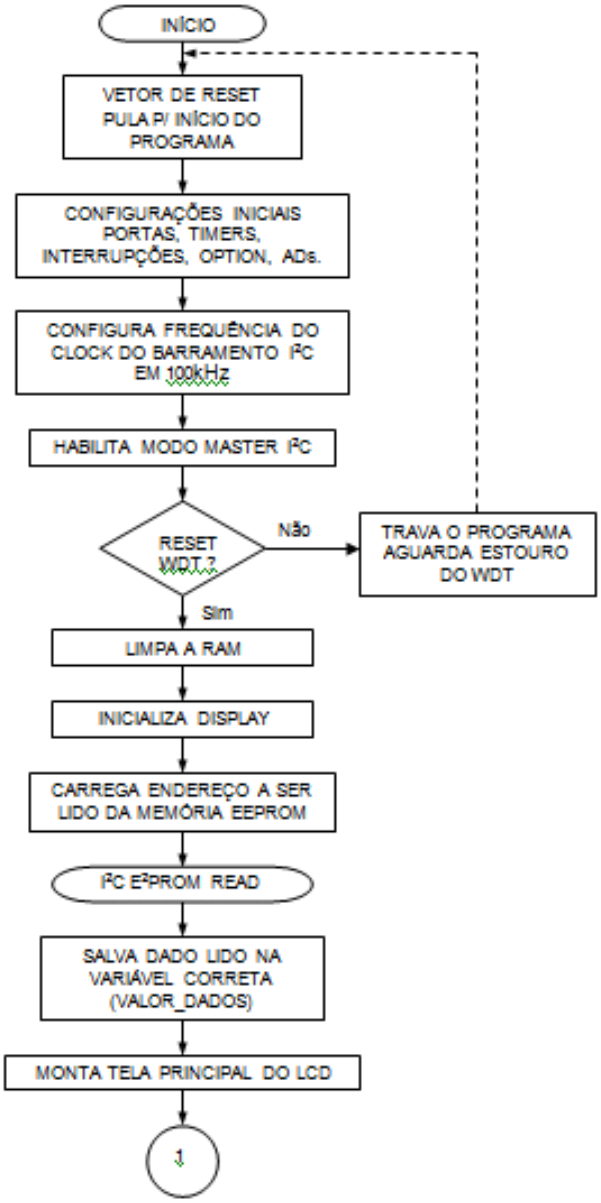
- Envia outro start bit;
- Envia novamente o byte de controle, porém agora alterando o bit R/W para sinalizar a operação de leitura. Assim, o byte de controle completo fica 10101111b;
- Recebe o bit de acknowledge (ACK);
- Recebe o byte lido da memória;
- Envia um bit em 1 sinalizando que deseja encerrar leitura (sinal de NACK);
- Envia um stop bit.

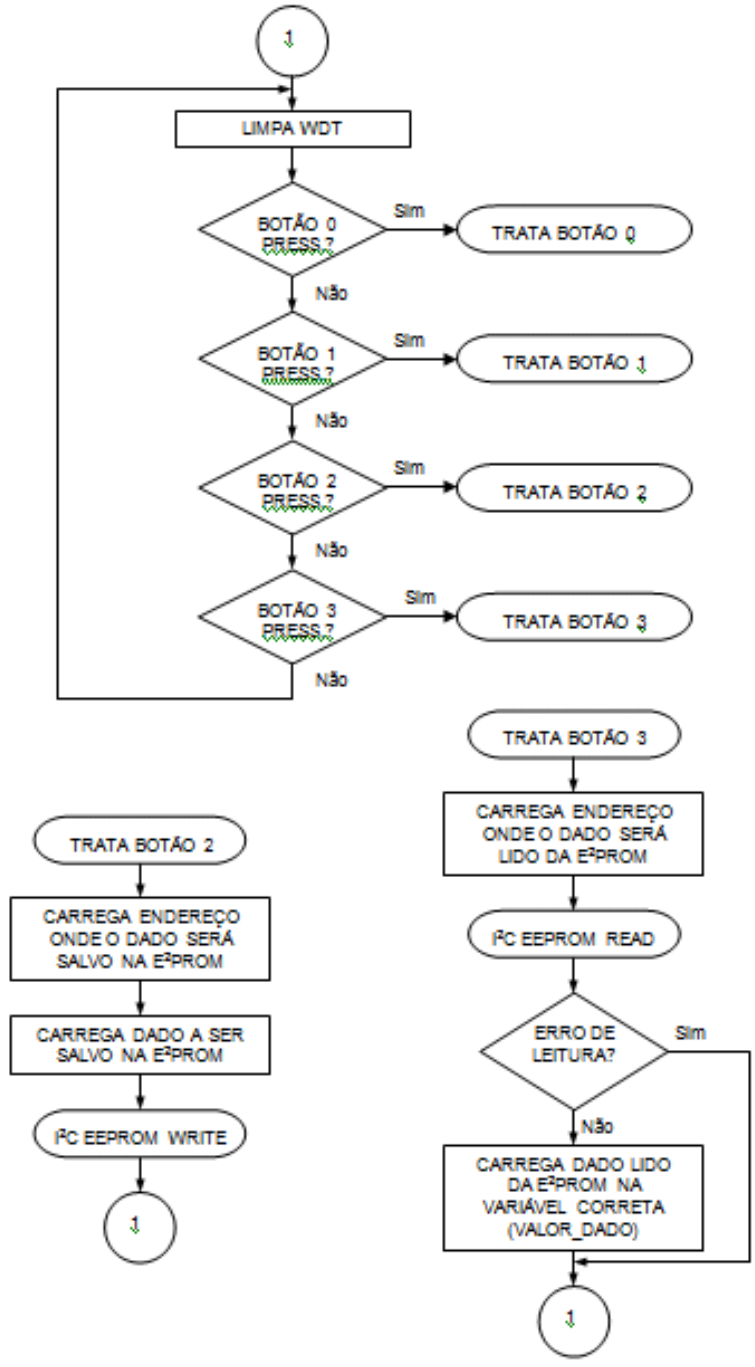
O exemplo da experiência foi elaborado utilizando os procedimentos descritos acima. São utilizados os botões da linha 4 para manipular um valor de 8 bits mostrado no display LCD. Este valor pode ser salvo e lido na memória EEPROM. Os botões das colunas 1 e 2 são utilizados para incrementar e decrementar o valor mostrado no display. O botão da coluna 3 salva o valor do display na memória serial enquanto o botão da coluna 4 é utilizado para ler o valor salvo na memória serial.

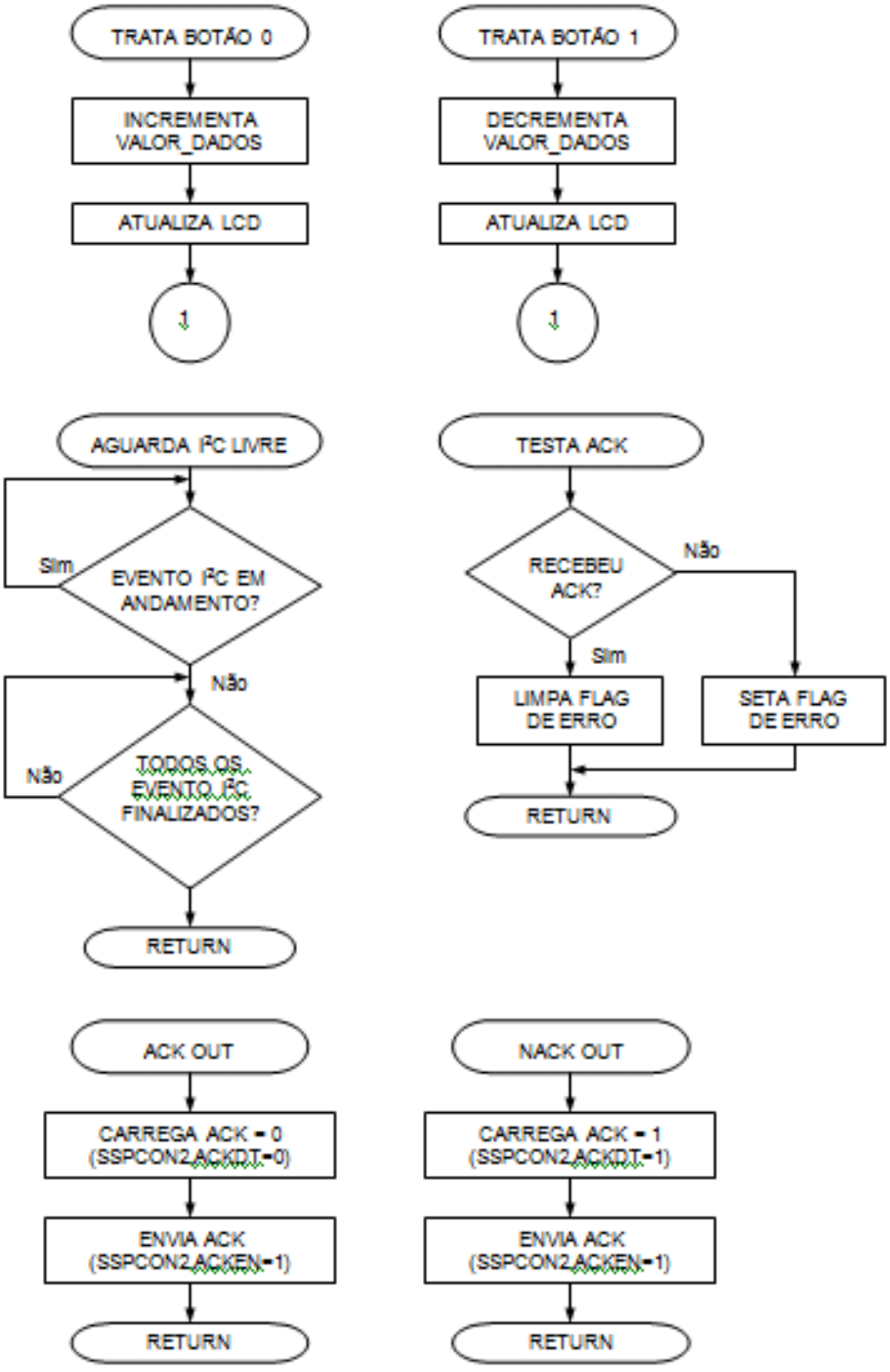
18.3. Esquema Elétrico

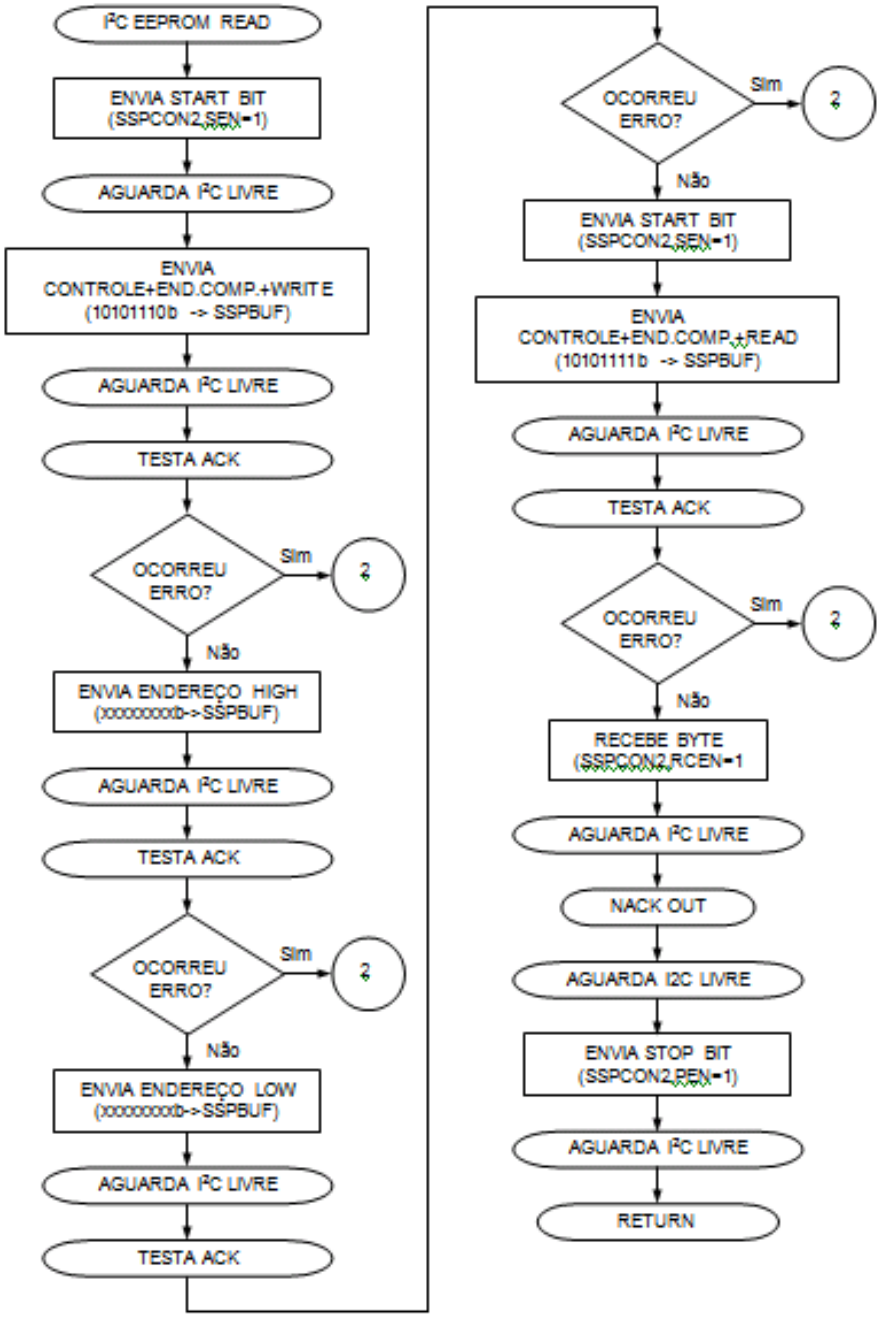


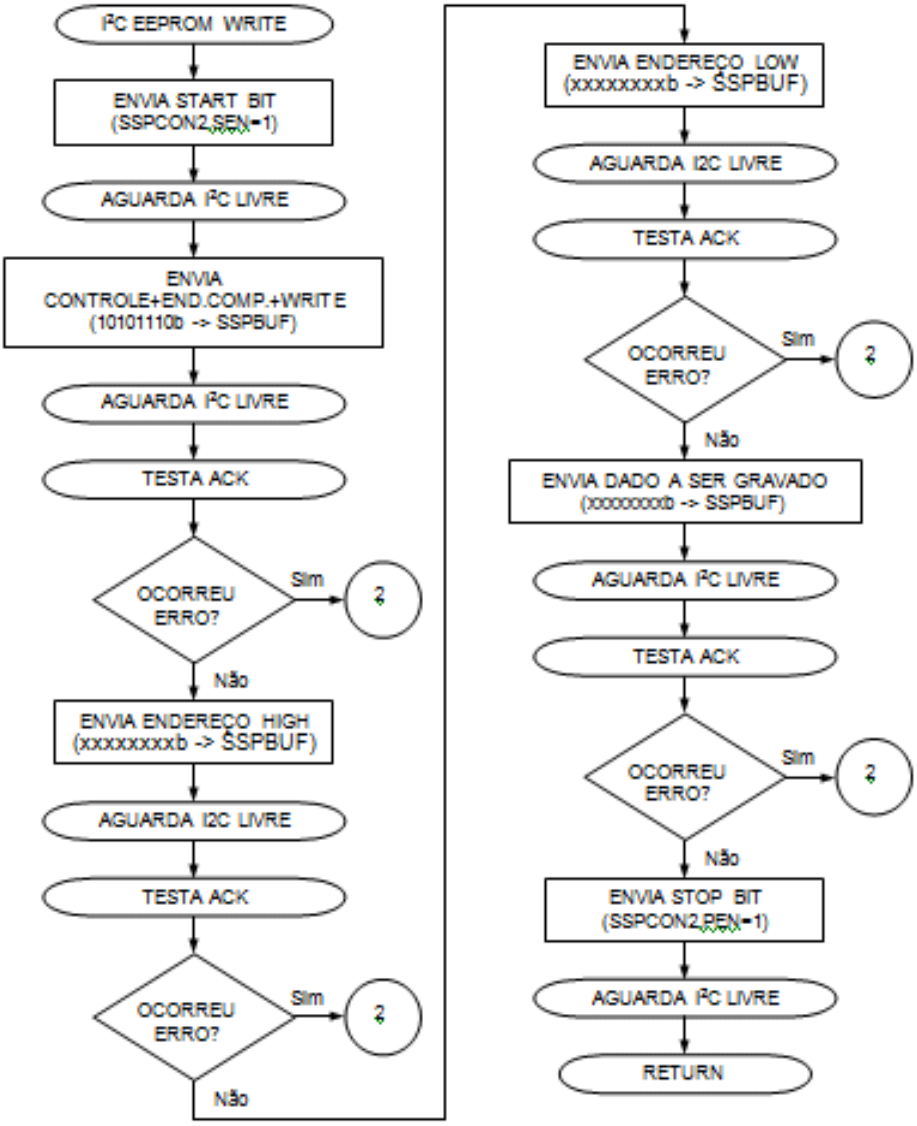
18.4. Fluxograma

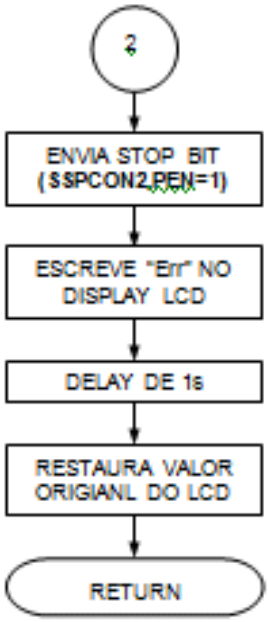

INÍCIO
VETOR DE RESET PULA P/ INÍCIO DO PROGRAMA
CONFIGURAÇÕES INICIAIS PORTAS, TIMERS, INTERRUPÇÕES, OPTION, ADs.
CONFIGURA FREQUÊNCIA DO CLOCK DO BARRAMENTO I²C EM 100kHz
HABILITA MODO MASTER I²C
RESET WDT ?
Não
TRAVA O PROGRAMA AGUARDA ESTOURO DO WDT
Sim
LIMPA A RAM
INICIALIZA DISPLAY
CARREGA ENDEREÇO A SER LIDO DA MEMÓRIA EEPROM
I²C E²PROM READ
SALVA DADO LIDO NA VARIÁVEL CORRETA (VALOR_DADOS)
MONTA TELA PRINCIPAL DO LCD
1


1
LIMPA WDT
BOTÃO 0 PRESS.?
Sim
TRATA BOTÃO 0
Não
BOTÃO 1 PRESS.?
Sim
TRATA BOTÃO 1
Não
BOTÃO 2 PRESS.?
Sim
TRATA BOTÃO 2
Não
BOTÃO 3 PRESS.?
Sim
TRATA BOTÃO 3
Não
TRATA BOTÃO 2
CARREGA ENDEREÇO ONDE O DADO SERÁ SALVO NA E²PROM
CARREGA DADO A SER SALVO NA E²PROM
I²C EEPROM WRITE
1
TRATA BOTÃO 3
CARREGA ENDEREÇO ONDE O DADO SERÁ LIDO DA E²PROM
I²C EEPROM READ
ERRO DE LEITURA?
Sim
Não
CARREGA DADO LIDO DA E²PROM NA VARIÁVEL CORRETA (VALOR_DADO)
1


TRATA BOTÃO 0
INCREMENTA VALOR_DADOS
ATUALIZA LCD
1
TRATA BOTÃO 1
DECREMENTA VALOR_DADOS
ATUALIZA LCD
1
AGUARDA I²C LIVRE
Sim
EVENTO I²C EM ANDAMENTO?
Não
Não
TODOS OS EVENTO I²C FINALIZADOS?
RETURN
TESTA ACK
RECEBEU ACK?
Não
Sim
LIMPA FLAG DE ERRO
SETA FLAG DE ERRO
RETURN
ACK OUT
CARREGA ACK = 0 (SSPCON2.ACKDT=0)
ENVIA ACK (SSPCON2.ACKEN=1)
RETURN
NACK OUT
CARREGA ACK = 1 (SSPCON2.ACKDT=1)
ENVIA ACK (SSPCON2.ACKEN=1)
RETURN


I²C EEPROM READ
ENVIA START BIT (SSPCON2.SEN=1)
AGUARDA I²C LIVRE
ENVIA CONTROLE+END.COMP.+WRITE (10101110b -> SSPBUF)
AGUARDA I²C LIVRE
TESTA ACK
OCORREU ERRO?
Sim
2
Não
ENVIA ENDEREÇO HIGH (xxxxxxxxb->SSPBUF)
AGUARDA I²C LIVRE
TESTA ACK
OCORREU ERRO?
Sim
2
Não
ENVIA ENDEREÇO LOW (xxxxxxxxb->SSPBUF)
AGUARDA I²C LIVRE
TESTA ACK
OCORREU ERRO?
Sim
2
Não
ENVIA START BIT (SSPCON2.SEN=1)
ENVIA CONTROLE+END.COMP.+READ (10101111b -> SSPBUF)
AGUARDA I²C LIVRE
TESTA ACK
OCORREU ERRO?
Sim
2
Não
RECEBE BYTE (SSPCON2.RCEN=1
AGUARDA I²C LIVRE
NACK OUT
AGUARDA I2C LIVRE
ENVIA STOP BIT (SSPCON2.PEN=1)
AGUARDA I²C LIVRE
RETURN


I²C EEPROM WRITE
ENVIA START BIT (SSPCON2.SEN=1)
AGUARDA I²C LIVRE
ENVIA CONTROLE+END.COMP.+WRITE (10101110b -> SSPBUF)
AGUARDA I²C LIVRE
TESTA ACK
OCORREU ERRO?
Sim
2
Não
ENVIA ENDEREÇO HIGH (xxxxxxxxb -> SSPBUF)
AGUARDA I2C LIVRE
TESTA ACK
OCORREU ERRO?
Sim
2
Não
ENVIA ENDEREÇO LOW (xxxxxxxxb -> SSPBUF)
AGUARDA I2C LIVRE
TESTA ACK
OCORREU ERRO?
Sim
2
Não
ENVIA DADO A SER GRAVADO (xxxxxxxxb -> SSPBUF)
AGUARDA I²C LIVRE
TESTA ACK
OCORREU ERRO?
Sim
2
Não
ENVIA STOP BIT (SSPCON2.PEN=1)
AGUARDA I²C LIVRE
RETURN


2
ENVIA STOP BIT
(SSPCON2.PEN=1)
ESCREVE "Err" NO
DISPLAY LCD
DELAY DE 1s
RESTAURA VALOR
ORIGIANL DO LCD
RETURN

18.5. Código

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Exemplo de Código para utilização com a McMaster            *
 *                              Exemplo 16                                 *
 *                                                                         *
 *                               MOSAICO                                   *
 *                                                                         *
 *  TEL: (0XX11) 4992-8775              SITE: www.mosaico.com.br           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                          *
 *   DATA : 18/01/2010                                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Descrição geral                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

//ESTE EXEMPLO FOI ELABORADO PARA EXPLICAR O FUNCIONAMENTO DA LEITURA/ESCRITA
//NA MEMÓRIA E2PROM SERIAL EXTERNA, UTILIZANDO O MASTER I2C.
//AS FUNÇÕES DOS BOTÕES NA MATRIZ SÃO:
//                            COLUNA 3        COLUNA4
//            LINHA4 ->    DECREMENTA VALOR     INCREMENTA VALOR
//            LINHA3 ->    ESCRITA DO VALOR     LEITURA DO VALOR
/*
 CONFIGURAÇÂO DOS JUMPERS DA PLACA:

        CAP USB(RC3)              -> ON
        DATA I2C                  -> ON
        CLOCK I2C                 -> ON
        COLUNA 2(RB1)             -> OFF
        COLUNA 1(RB0)             -> OFF
        LINHA 1 / DISPLAY MILHAR(RB4)    -> ON
        LEDS ESPECIAS (RC0,RC1,RC2)      -> ON

*/
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            DEFINIÇÃO PIC                                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <p18F4550.h>           // Register definitions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               INCLUDES DAS FUNÇÕES DE PERIFÉRICOS DO PIC                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <pwm.h>                //PWM library functions
#include <adc.h>                //ADC library functions
#include <timers.h>             //Timer library functions
#include <delays.h>             //Delay library functions
#include <i2c.h>                //I2C library functions
#include <stdlib.h>             //Library functions
#include <usart.h>              //USART library functions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Configurações para gravação                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#pragma config FOSC = HS
#pragma config CPUDIV = OSC1_PLL2
```

```
#pragma config WDT = ON
#pragma config WDTPS = 128
#pragma config LVP = OFF
#pragma config PWRT = ON
#pragma config BOR = OFF
#pragma config BORV = 0
#pragma config PBADEN = OFF
#pragma config DEBUG = OFF
#pragma config PLLDIV = 1
#pragma config USBDIV = 1
#pragma config FCMEN = OFF
#pragma config IESO = OFF
#pragma config VREGEN = OFF
#pragma config MCLRE = ON
#pragma config LPT1OSC = OFF
#pragma config CCP2MX = ON
#pragma config STVREN = OFF
#pragma config ICPRT = OFF

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Definição e inicialização das variáveis Globais          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.
unsigned char    FILTRO11;        //FILTRAGEM  1 PARA O BOTÃO 1
unsigned char    FILTRO12;        //FILTRAGEM  2 PARA O BOTÃO 1
unsigned char    FILTRO21;        //FILTRAGEM  1 PARA O BOTÃO 2
unsigned char    FILTRO22;        //FILTRAGEM  2 PARA O BOTÃO 2
unsigned char    FILTRO31;        //FILTRAGEM  1 PARA O BOTÃO 3
unsigned char    FILTRO32;        //FILTRAGEM  2 PARA O BOTÃO 3
unsigned char    FILTRO41;        //FILTRAGEM  1 PARA O BOTÃO 4
unsigned char    FILTRO42;        //FILTRAGEM  2 PARA O BOTÃO 4

unsigned char    valor_dados = 0;

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Constantes internas                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de constantes facilita a programação e a manutenção.
#define T_FILTRO    25

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Declaração dos flags de software                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de flags ajuda na programação e economiza memória RAM.
struct
{
    unsigned BIT0:1;
    unsigned BIT1:1;
    unsigned BIT2:1;
    unsigned BIT3:1;
    unsigned BIT4:1;
    unsigned BIT5:1;
    unsigned BIT6:1;
    unsigned BIT7:1;
}FLAGSbits;             //ARMAZENA OS FLAGS DE CONTROLE

#define    ST_BT1        FLAGSbits.BIT0        //STATUS DO BOTÃO 1
#define    ST_BT2        FLAGSbits.BIT1        //STATUS DO BOTÃO 2
#define    ST_BT3        FLAGSbits.BIT2        //STATUS DO BOTÃO 3
#define    ST_BT4        FLAGSbits.BIT3        //STATUS DO BOTÃO 4
#define    ST_I2C        FLAGSbits.BIT4        //STATUS DO I2C

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Definição e inicialização dos port's                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      PROTOTIPAGEM DE FUNÇÕES                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void comando_lcd(unsigned char caracter);
void escreve_lcd(unsigned char caracter);
void limpa_lcd(void);
void inicializa_lcd(void);
void tela_principal(void);
void I2C_EEPROM_WRITE(unsigned int endereco, unsigned char dado);
unsigned char I2C_EEPROM_READ(unsigned int endereco);
void atualiza_linha_1(void);
void atualiza_linha_2(unsigned char aux);

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                               ENTRADAS                                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//As entradas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define     BT1     PORTBbits.RB2   //PORTA DO BOTÃO
                                    //1 -> PRESSIONADO
                                    //0 -> LIBERADO

#define     BT2     PORTBbits.RB3   //PORTA DO BOTÃO
                                    //1 -> PRESSIONADO
                                    //0 -> LIBERADO

#define     BT3     PORTBbits.RB2   //PORTA DO BOTÃO
                                    //1 -> PRESSIONADO
                                    //0 -> LIBERADO

#define     BT4     PORTBbits.RB3   //PORTA DO BOTÃO
                                    //1 -> PRESSIONADO
                                    //0 -> LIBERADO

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              SAÍDAS                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//As saídas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define     LINHA_4     LATBbits.LATB7   //PINO PARA ATIVAR LINHA 4 DO TECLADO
                                        //MATRICIAL
                                        //1 -> LINHA ATIVADA
                                        //0 -> LINHA DESATIVADA

#define     LINHA_3     LATBbits.LATB6   //PINO PARA ATIVAR LINHA 4 DO TECLADO
                                        //MATRICIAL
                                        //1 -> LINHA ATIVADA
                                        //0 -> LINHA DESATIVADA

#define     rs      LATEbits.LATE0       // via do lcd que sinaliza recepção de dados
ou comando
#define     enable  LATEbits.LATE1       // enable do lcd

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Rotina que envia um COMANDO para o LCD               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void comando_lcd(unsigned char caracter)
{
    rs = 0;                     // seleciona o envio de um comando
    PORTD = caracter;           // carrega o PORTD com o caracter
    enable = 1 ;                // gera pulso no enable
    Delay10TCYx(1);             // espera 3 microsegundos
    enable = 0;                 // desce o pino de enable
```

```
    Delay10TCYx(4);             // espera mínimo 40 microsegundos
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Rotina que envia um DADO a ser escrito no LCD              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void escreve_lcd(unsigned char caracter)
{
    rs = 1;                     // seleciona o envio de um comando
    PORTD = caracter;           // carrega o PORTD com o caracter
    enable = 1;                 // gera pulso no enable
    Delay10TCYx(1);             // espera 10 microsegundos
    enable = 0;                 // desce o pino de enable
    Delay10TCYx(4);             // espera mínimo 40 microsegundos
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            Função para limpar o LCD                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void limpa_lcd(void)
{
    comando_lcd(0x01);          // limpa lcd
    Delay1KTCYx(2);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Inicialização do Display de LCD                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void inicializa_lcd(void)
{
    comando_lcd(0x30);          // envia comando para inicializar display
    Delay1KTCYx(4);             // espera 4 milisengundos

    comando_lcd(0x30);          // envia comando para inicializar display
    Delay10TCYx(10);            // espera 100 microsengundos

    comando_lcd(0x30);          // envia comando para inicializar display

    comando_lcd(0x38);          // liga o display, sem cursor e sem blink

    limpa_lcd();                // limpa lcd

    comando_lcd(0x0c);          // display sem cursor

    comando_lcd(0x06);          // desloca cursor para a direita
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Tela Principal                                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void atualiza_linha_1(void)
{
    comando_lcd(0x83);      // posiciona o cursor na linha 0, coluna 0
    escreve_lcd ('M');      // imprime mensagem no lcd
    escreve_lcd ('A');
    escreve_lcd ('S');
    escreve_lcd ('T');
    escreve_lcd ('E');
    escreve_lcd ('R');
    escreve_lcd (' ');
    escreve_lcd ('I');
    escreve_lcd ('2');
```

```
    escreve_lcd ('C');
}


void atualiza_linha_2(unsigned char aux)
{
    unsigned char hexa_high;
    unsigned char hexa_low;

    hexa_low = aux & 0b00001111;
    hexa_high = (aux >> 4) & 0b00001111;

    comando_lcd(0xC6);      // posiciona o cursor na linha 0, coluna 0
    if(hexa_high <= 9)    escreve_lcd (hexa_high + 0x30);  // imprime mensagem no lcd
    else escreve_lcd(hexa_high + 0x37);

    if(hexa_low <= 9)    escreve_lcd (hexa_low + 0x30);    // imprime mensagem no lcd
    else escreve_lcd(hexa_low + 0x37);

    escreve_lcd ('h');      // imprime mensagem no lcd
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              ESCRITA DA EEPROM                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                Função para escrita na EEPROM externa I2C                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void I2C_EEPROM_WRITE(unsigned int endereco, unsigned char dado)
{
    unsigned char endereco_high;
    unsigned char endereco_low;

    endereco_high = endereco >> 8;
    endereco_low = endereco;

    StartI2C();
    while(SSPCON2bits.SEN);
    WriteI2C(0b10101110);       // controle de escrita
    IdleI2C();                  // espera fim do evento I2C
    WriteI2C(endereco_high);
    IdleI2C();                  // espera fim do evento I2C
    WriteI2C(endereco_low);
    IdleI2C();                  // espera fim do evento I2C
    WriteI2C(dado);             // controle de leitura
    StopI2C();                  // fim de comunicação
    while(SSPCON2bits.PEN);
    ClrWdt();
    Delay100TCYx(100);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                Função para leitura na EEPROM externa I2C                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

unsigned char I2C_EEPROM_READ(unsigned int endereco)
{
    unsigned char endereco_high;
    unsigned char endereco_low;
    unsigned char dado;

    endereco_high = endereco >> 8;
    endereco_low = endereco;
```

```
    StartI2C();
    while(SSPCON2bits.SEN);
    WriteI2C(0b10101110);       // controle de escrita
    IdleI2C();                  // espera fim do evento I2C
    WriteI2C(endereco_high);
    IdleI2C();                  // espera fim do evento I2C
    WriteI2C(endereco_low);
    IdleI2C();                  // espera fim do evento I2C
    RestartI2C();
    while(SSPCON2bits.SEN);
    WriteI2C(0b10101111);       // controle de leitura
    IdleI2C();                  // espera fim do evento I2C
    SSPCON2bits.RCEN = 1;
    while(SSPCON2bits.RCEN);
    NotAckI2C();
    while(SSPCON2bits.ACKEN);
    StopI2C();                  // fim de comunicação
    while(SSPCON2bits.PEN);

    return((unsigned char) SSPBUF);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função Principal                             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void main ()
{

    PORTA = 0x00;               //Limpa PORTA
    PORTB = 0x00;               //Limpa PORTB
    PORTC = 0x00;               //Limpa PORTC
    PORTD = 0x00;               //Limpa PORTD
    PORTE = 0x00;               //Limpa PORTE

    LATA = 0x00;                //Limpa PORTA
    LATB = 0x00;                //Limpa PORTB
    LATC = 0x00;                //Limpa PORTC
    LATD = 0x00;                //Limpa PORTD
    LATE = 0x00;                //Limpa PORTE

    TRISA = 0b00101111;         //CONFIG DIREÇÃO DOS PINOS PORTA
    TRISB = 0b00001111;         //CONFIG DIREÇÃO DOS PINOS PORTB
    TRISC = 0b10010000;         //CONFIG DIREÇÃO DOS PINOS PORTC
    TRISD = 0b00000000;         //CONFIG DIREÇÃO DOS PINOS PORTD
    TRISE = 0b00000000;         //CONFIG DIREÇÃO DOS PINOS PORTE

    ADCON1 = 0b00001111;        //DESLIGA CONVERSORES A/D

    OpenI2C(MASTER,SLEW_ON);
    SSPADD = 0b00001001;

    while(RCONbits.NOT_TO);     //aguarda estouro do WDT

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Inicialização do Sistema                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

    RCON = 0x00;

    inicializa_lcd();           // configura o lcd

    atualiza_linha_1();
    valor_dados = I2C_EEPROM_READ(0x0100);
    atualiza_linha_2(valor_dados);
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Rotina Principal                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
    while(1)
    {
        ClrWdt();

        FILTRO11 = 0;           //BT1
        FILTRO12 = T_FILTRO;    //BT1

        FILTRO21 = 0;           //BT2
        FILTRO22 = T_FILTRO;    //BT2

        FILTRO31 = 0;           //BT3
        FILTRO32 = T_FILTRO;    //BT3

        FILTRO41 = 0;           //BT4
        FILTRO42 = T_FILTRO;    //BT4

        LINHA_4 = 1;                //ATIVA LINHA 4 DO TECLADO MATRICIAL
        LINHA_3 = 0;                //DESATIVA LINHA 3 DO TECLADO MATRICIAL


        while(BT1)
        {

            ClrWdt();
            if (--FILTRO11 == 0)
            {
                FILTRO12--;
                if (FILTRO12 == 0)
                {
                    if (!ST_BT1)
                    {
                        //função
                        valor_dados++;
                        atualiza_linha_2(valor_dados);
                        ST_BT1 = 1;
                    }
                }
            }
        }
        ST_BT1 = 0; // BOTÃO LIBERADO, LIMPA O FLAG

        while(BT2)
        {
            ClrWdt();
            if (--FILTRO21 == 0)
            {
                FILTRO22--;
                if (FILTRO22 == 0)
                {
                    if (!ST_BT2)
                    {
                        //função
                        valor_dados--;
                        atualiza_linha_2(valor_dados);
                        ST_BT2 = 1;
                    }
                }
            }
        }
        ST_BT2 = 0; // BOTÃO LIBERADO, LIMPA O FLAG

        LINHA_4 = 0;                //DESATIVA LINHA 4 DO TECLADO MATRICIAL
        LINHA_3 = 1;                //ATIVA LINHA 3 DO TECLADO MATRICIAL
```

```
        while(BT3)
        {
            ClrWdt();
            if (--FILTRO31 == 0)
            {
                FILTRO32--;
                if (FILTRO32 == 0)
                {
                    if (!ST_BT3)
                    {
                        //função
                        I2C_EEPROM_WRITE(0x0100,valor_dados);
                        atualiza_linha_2(valor_dados);
                        ST_BT3 = 1;
                    }
                }
            }
        }
        ST_BT3 = 0; // BOTÃO LIBERADO, LIMPA O FLAG


        while(BT4)
        {
            ClrWdt();
            if (--FILTRO41 == 0)
            {
                FILTRO42--;
                if (FILTRO42 == 0)
                {
                    if (!ST_BT4)
                    {
                        //função
                        valor_dados = I2C_EEPROM_READ(0x0100);
                        atualiza_linha_2(valor_dados);
                        ST_BT4 = 1;
                    }
                }
            }
        }
        ST_BT4 = 0; // BOTÃO LIBERADO, LIMPA O FLAG
    }
}// FIM LAÇO PRINCIPAL
```

18.6. Dicas e Comentários

As constantes END_EEPROM_H e END_EEPROM_L representam a posição a ser gravada/lida da memória externa.

Note que este programa não utiliza as interrupções de leitura e escrita relacionadas ao protocolo I^2C. Desta forma, ele possui uma rotina (AGUARDA_I2C_LIVRE) para saber se o sistema está liberado para a próxima ação. Ele também testa o ACK e gera uma mensagem de erro caso alguma coisa saia fora do padrão.

18.7. Exercícios Propostos

1. Faça três modificações no primeiro exercício proposto da experiência 15.
   - Utilize a memória externa;
   - Limite os dados mostrados no display entre 0x41 e 0x5A;
   - Mostre os dados em ASCII, ou seja, entre A (0x41) e Z (0x5A);
2. Utilizando o exercício anterior grave na memória uma mensagem de até 16 caracteres. Depois, crie um programa que ao ser inicializado leia os 16 caracteres da memória e mostre a mensagem lida no LCD;

19. Experiência 17 – Comunicação serial RS232 via USART

19.1. Objetivo

O objetivo desta experiência é o aprendizado do módulo USART do microcontrolador PIC18F4550 utilizado para implementar a comunicação padrão RS232, geralmente utilizada para estabelecer um canal de comunicação entre um microcontrolador e um computador.

19.2. Descrição

Para tornar o sistema versátil e simples, criou-se um programa capaz de testar a transmissão e recepção de dados de modo isolado, ou seja, apenas com o McMaster sem necessariamente conecta-lo ao computador. Embora nada impeça que a comunicação com o PC seja efetivamente realizada.

Para atender esta necessidade, o software da experiência implementa uma comunicação assíncrona *Full duplex*, isto é, com a transmissão e a recepção ativadas simultaneamente.

O valor transmitido é obtido a partir da leitura da tensão do potenciômetro através do conversor A/D limitando este valor entre 0 e 255 (8-bits). O valor do A/D é então enviado para a porta serial e para o LCD. Desta forma é possível visualizar o dado transmitido. Para facilitar ainda mais o usuário, mostra-se o valor em decimal (d) e em hexadecimal (h). A comunicação é realizada no padrão 8N1 com uma velocidade de 9.600bps.

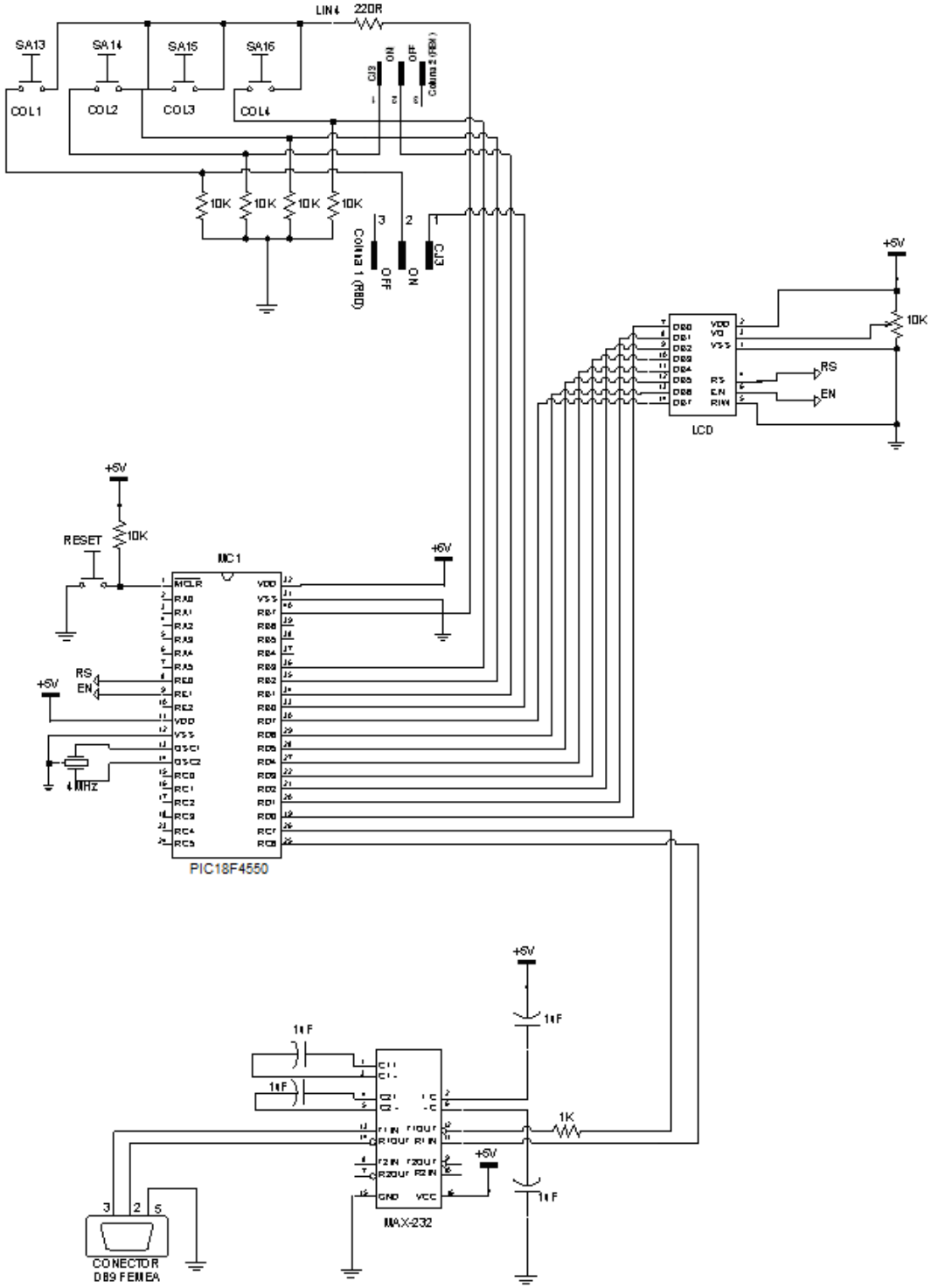
Quanto à recepção, o valor obtido pela porta serial é diretamente impresso no display de LCD, através do código ASCII.

Para que o sistema funcione sem o PC, basta interligar os pinos 2 e 3 do conector DB9. Isto fará com que tudo que seja transmitido por TX seja imediatamente recebido em RX. Tanto a transmissão quanto a recepção são contínuas.

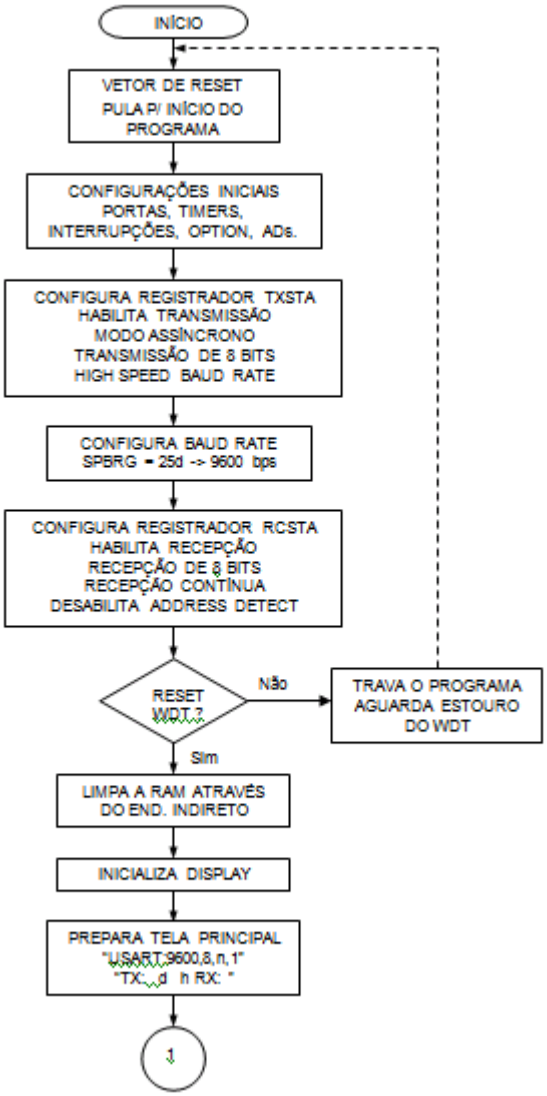
Não se deve esquecer de habilitar a porta RS232 para o microcontrolador através do botão de modo de operação após.

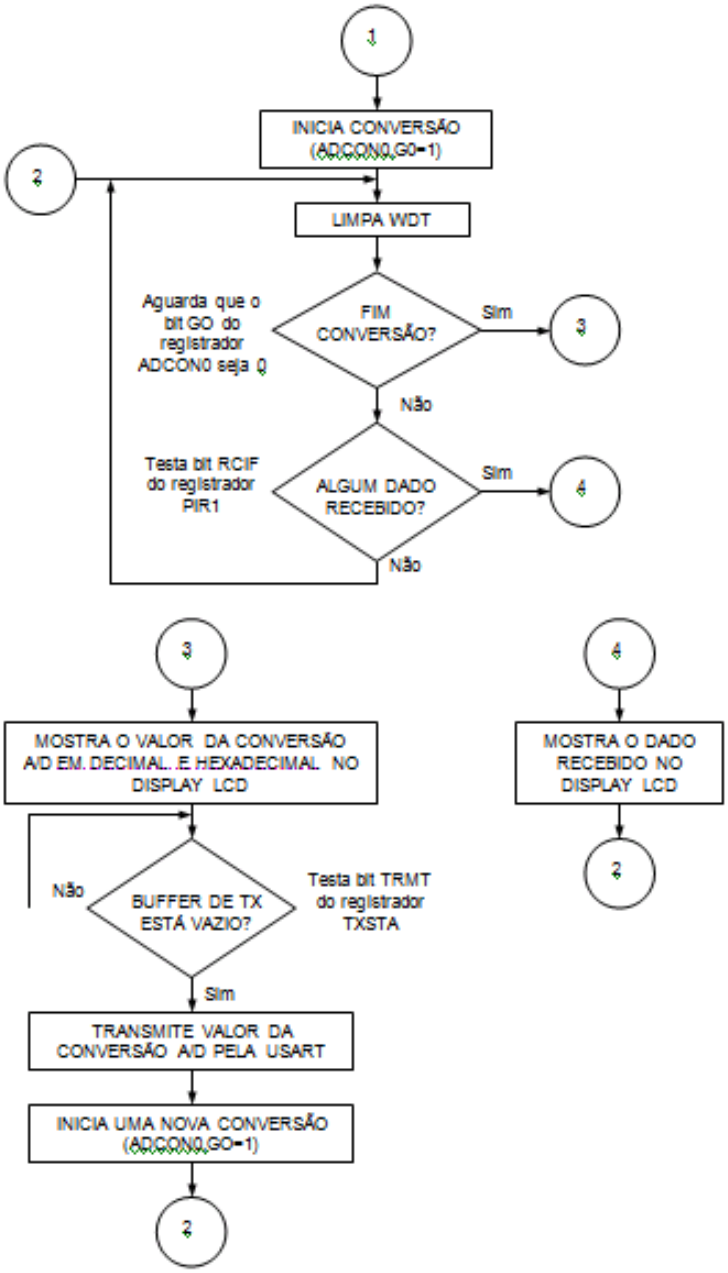
Para testar a comunicação com o computador (PC) pode-se utilizar o software SDCom disponível no CD.

19.3. Esquema Elétrico



19.4. Fluxograma




1
INICIA CONVERSÃO (ADCON0,GO=1)
2
LIMPA WDT
Aguarda que o bit GO do registrador ADCON0 seja 0
FIM CONVERSÃO?
Sim
3
Não
Testa bit RCIF do registrador PIR1
ALGUM DADO RECEBIDO?
Sim
4
Não
3
MOSTRA O VALOR DA CONVERSÃO A/D EM DECIMAL E HEXADECIMAL NO DISPLAY LCD
Não
BUFFER DE TX ESTÁ VAZIO?
Testa bit TRMT do registrador TXSTA
Sim
TRANSMITE VALOR DA CONVERSÃO A/D PELA USART
INICIA UMA NOVA CONVERSÃO (ADCON0,GO=1)
2
4
MOSTRA O DADO RECEBIDO NO DISPLAY LCD
2

19.5. Código

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Exemplo de Código para utilização com a McMaster            *
 *                              Exemplo 17                                 *
 *                                                                         *
 *                               MOSAICO                                   *
 *                                                                         *
 *  TEL: (0XX11) 4992-8775                SITE: www.mosaico.com.br         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                          *
 *   DATA : 18/01/2010                                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Descrição geral                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*
ESTE EXEMPLO FOI ELABORADO PARA EXPLICAR O FUNCIONAMENTO DA USART DO PIC.
O SOFTWARE CONVERTE O CANAL 1 DO CONVERSOR A/D (POTENCIÔMETRO P2) E MOSTRA
NO DISPLAY O VALOR CONVERTIDO EM DECIMAL E HEXADECIMAL.
ALÉM DE MOSTRAR O VALOR NO DISPLAY, O SOFTWARE TRANSMITE PELA USART O VALOR
DA CONVERSÃO. OS VALORES RECEBIDOS PELA USART TAMBÉM SÃO MOSTRADOS NO LCD
COMO CARACTERES ASCII.

*/

/*      CONFIGURAÇÂO DOS JUMPERS DA PLACA:

        CAP USB(RC3)                -> ON
        DATA I2C                    -> OFF
        CLOCK I2C                   -> OFF
        COLUNA 2(RB1)               -> ON
        COLUNA 1(RB0)               -> ON
        LINHA 1 / DISPLAY MILHAR(RB4)    -> ON
        LEDS ESPECIAS (RC0,RC1,RC2)      -> ON

*/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           DEFINIÇÃO PIC                                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <p18F4550.h>           // Register definitions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               INCLUDES DAS FUNÇÕES DE PERIFÉRICOS DO PIC                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <pwm.h>                //PWM library functions
#include <adc.h>                //ADC library functions
#include <timers.h>             //Timer library functions
#include <delays.h>             //Delay library functions
#include <i2c.h>                //I2C library functions
#include <stdlib.h>             //Library functions
#include <usart.h>              //USART library functions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Configurações para gravação                      *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
```

```
#pragma config FOSC = HS
#pragma config CPUDIV = OSC1_PLL2
#pragma config WDT = OFF
#pragma config WDTPS = 128
#pragma config LVP = OFF
#pragma config PWRT = ON
#pragma config BOR = OFF
#pragma config BORV = 0
#pragma config PBADEN = OFF
#pragma config DEBUG = OFF
#pragma config PLLDIV = 1
#pragma config USBDIV = 1
#pragma config FCMEN = OFF
#pragma config IESO = OFF
#pragma config VREGEN = OFF
#pragma config MCLRE = ON
#pragma config LPT1OSC = OFF
#pragma config CCP2MX = ON
#pragma config STVREN = OFF
#pragma config ICPRT = OFF

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Definição e inicialização das variáveis Globais          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.

unsigned char unidade;
unsigned char dezena;
unsigned char centena;
unsigned char hexa_low;
unsigned char hexa_high;

unsigned char converte;
char usart_rx;

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Constantes internas                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de constantes facilita a programação e a manutenção.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                    Declaração dos flags de software                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de flags ajuda na programação e economiza memória RAM.
//Este programa não utiliza nenhum flag de usuário

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Definição e inicialização dos port's                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        PROTOTIPAGEM DE FUNÇÕES                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void comando_lcd(unsigned char caracter);
void escreve_lcd(unsigned char caracter);
void limpa_lcd(void);
void inicializa_lcd(void);
void tela_principal(void);
void converte_bcd(unsigned char aux);
void converte_hexadecimal(unsigned char aux);

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              ENTRADAS                                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
```

```
// As entradas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              SAÍDAS                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//As saídas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define     rs      PORTEbits.RE0   // via do lcd que sinaliza recepção de dados ou
comando
#define     enable  PORTEbits.RE1   // enable do lcd

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Rotina que envia um COMANDO para o LCD              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void comando_lcd(unsigned char caracter)
{
    rs = 0;                     // seleciona o envio de um comando
    PORTD = caracter;           // carrega o PORTD com o caracter
    enable = 1 ;                // gera pulso no enable
    Delay10TCYx(1);             // espera 3 microsegundos
    enable = 0;                 // desce o pino de enable
    Delay10TCYx(4);             // espera mínimo 40 microsegundos
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               Rotina que envia um DADO a ser escrito no LCD           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void escreve_lcd(unsigned char caracter)
{
    rs = 1;                     // seleciona o envio de um comando
    PORTD = caracter;           // carrega o PORTD com o caracter
    enable = 1;                 // gera pulso no enable
    Delay10TCYx(1);             // espera 10 microsegundos
    enable = 0;                 // desce o pino de enable
    Delay10TCYx(4);             // espera mínimo 40 microsegundos
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            Função para limpar o LCD                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void limpa_lcd(void)
{
    comando_lcd(0x01);          // limpa lcd
    Delay1KTCYx(2);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Inicialização do Display de LCD                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void inicializa_lcd(void)
{
    comando_lcd(0x30);          // envia comando para inicializar display
    Delay1KTCYx(4);             // espera 4 milisengundos

    comando_lcd(0x30);          // envia comando para inicializar display
    Delay10TCYx(10);            // espera 100 microsengundos

    comando_lcd(0x30);          // envia comando para inicializar display

    comando_lcd(0x38);          // liga o display, sem cursor e sem blink
```

```
    limpa_lcd();                // limpa lcd

    comando_lcd(0x0c);          // display sem cursor

    comando_lcd(0x06);          // desloca cursor para a direita
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              Tela Principal                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void tela_principal(void)
{
    comando_lcd(0x80);     // posiciona o cursor na linha 0, coluna 0
    escreve_lcd ('U');     // imprime mensagem no lcd
    escreve_lcd ('S');
    escreve_lcd ('A');
    escreve_lcd ('R');
    escreve_lcd ('T');
    escreve_lcd (':');
    escreve_lcd ('9');
    escreve_lcd ('6');
    escreve_lcd ('0');
    escreve_lcd ('0');
    escreve_lcd (',');
    escreve_lcd ('8');
    escreve_lcd (',');
    escreve_lcd ('n');
    escreve_lcd (',');
    escreve_lcd ('1');

    comando_lcd(0xC0);     // posiciona o cursor na linha 1, coluna 4
    escreve_lcd ('T');     // imprime mensagem no lcd
    escreve_lcd ('X');
    escreve_lcd (':');
    escreve_lcd (' ');
    escreve_lcd (' ');
    escreve_lcd (' ');
    escreve_lcd ('d');
    escreve_lcd (' ');
    escreve_lcd (' ');
    escreve_lcd (' ');
    escreve_lcd ('h');
    escreve_lcd (' ');
    escreve_lcd ('R');
    escreve_lcd ('X');
    escreve_lcd (':');
    escreve_lcd (' ');

}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              Converte BCD                                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void converte_bcd(unsigned char aux)
{
    unidade = 0;
    dezena = 0;
    centena = 0;
    if (aux == 0)return;
    while(aux--)
    {
        unidade++;
        if(unidade != 10)continue;
```

```
        unidade = 0;
        dezena++;
        if (dezena != 10)continue;
        dezena = 0;
        centena++;
        if (centena != 10)continue;
        centena = 0;
    }
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Converte hexadecimal                         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void converte_hexadecimal(unsigned char aux)
{
    hexa_low = aux & 0b00001111;
    hexa_high = (aux >> 4) & 0b00001111;
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função Principal                             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void main ()
{

    PORTA = 0x00;               //Limpa PORTA
    PORTB = 0x00;               //Limpa PORTB
    PORTC = 0x00;               //Limpa PORTC
    PORTD = 0x00;               //Limpa PORTD
    PORTE = 0x00;               //Limpa PORTE

    LATA = 0x00;                //Limpa PORTA
    LATB = 0x00;                //Limpa PORTB
    LATC = 0x00;                //Limpa PORTC
    LATD = 0x00;                //Limpa PORTD
    LATE = 0x00;                //Limpa PORTE

    TRISA = 0b00101111;         //CONFIG DIREÇÃO DOS PINOS PORTA
    TRISB = 0b00001111;         //CONFIG DIREÇÃO DOS PINOS PORTB
    TRISC = 0b10011111;         //CONFIG DIREÇÃO DOS PINOS PORTC
    TRISD = 0b00000000;         //CONFIG DIREÇÃO DOS PINOS PORTD
    TRISE = 0b00000000;         //CONFIG DIREÇÃO DOS PINOS PORTE

    OpenADC(ADC_FOSC_8 & ADC_LEFT_JUST & ADC_0_TAD, ADC_CH0 & ADC_INT_OFF &
ADC_REF_VDD_VSS,12);
    //CONFIGURAÇÃO DO AD
    OpenUSART(USART_TX_INT_OFF & USART_RX_INT_OFF & USART_ASYNCH_MODE &
              USART_EIGHT_BIT  & USART_CONT_RX    & USART_BRGH_HIGH,25);
    //CONFIGURAÇÃO DA USART

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Inicialização do Sistema                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

    RCON = 0x00;

    inicializa_lcd();           // configura o lcd

    tela_principal();

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Rotina Principal                             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
    ConvertADC();               //Inicia conversão AD
```

```
    while(1)
    {

        if(BusyADC())
        {
            converte = ADRESH;
            converte_bcd(converte);
            comando_lcd(0xC3);              // posiciona o cursor na linha 1, coluna 4
            escreve_lcd (centena + 0x30);   // imprime mensagem no lcd
            escreve_lcd (dezena + 0x30);
            escreve_lcd (unidade + 0x30);

            converte_hexadecimal(ADRESH);
            comando_lcd(0xC8);              // posiciona o cursor na linha 1, coluna 4
            if(hexa_high <= 9)    escreve_lcd (hexa_high + 0x30);  // imprime mensagem
no lcd
            else escreve_lcd(hexa_high + 0x37);

            if(hexa_low <= 9)    escreve_lcd (hexa_low + 0x30);    // imprime mensagem
no lcd
            else escreve_lcd(hexa_low + 0x37);

              while(BusyUSART());       //AGUARDA O BUFFER DE TRANSMISSÃO FICAR VAZIO

            WriteUSART(converte);
              ConvertADC();                   //Inicia conversão AD
        }

        if(DataRdyUSART())
        {
            usart_rx = ReadUSART();
            comando_lcd(0xCF);              // posiciona o cursor na linha 1, coluna 4
            escreve_lcd(usart_rx);

            ConvertADC();                   //Inicia conversão AD
        }
    }// FIM LAÇO PRINCIPAL
```

19.6. Dicas e Comentários

A rotina de conversão Hex >>> Decimal deste exemplo é mais completa do que nos casos anteriores, pois trabalha com 3 dígitos (CENTENA, DEZENA e UNIDADE). Desta forma, ela pode converter todo o *range* do argumento de entrada (W) que vai de 0 a 255.

O sistema de conversão A/D é o mesmo apresentado na experiência 11, onde utilizou-se o conversor interno considerando apenas os 8 bits mais significativos. Com isso o valor a ser transmitido já fica limitado a um byte.

Devido a simplicidade do sistema não foi necessário o uso das interrupções, deixando-as desabilitadas. Para o caso da recepção o bit **RCIF** é testando toda vez dentro do *loop* principal. Quanto a transmissão, sempre que um novo valor foi convertido, checa-se se o *buffer* de saída está vazio para poder escrever o novo valor.

19.7. Exercícios Propostos

1. Ative o uso da interrupção de recebimento. Quanto à transmissão, em vez de deixá-la contínua, crie uma interrupção de timer como base de tempo. Por exemplo, transmita o valor atual convertido a cada 1 segundo;
2. Crie um programa no PC que receba o valor convertido, efetue alguma operação e devolva outro valor. Por exemplo, divida o valor por 25, pegue a parte inteira e some 30h para imprimir no LCD um valor de 0 a 9;
3. Mude a rotina de recepção e escrita no LCD para poder receber um número de 0 a 50 e mostrá-lo como 0.0 a 5.0. Altere o programa do PC para efetuar a regra de 3 necessária para converter um valor de 0 a 255 para 0 a 50. Com isso você voltou ao multímetro da experiência 11, só que com as contas de multiplicação e divisão não mais sendo feitas no PIC.

20. Experiência 18 – Teclado matricial 4x4

20.1. Objetivo

O objetivo desta experiência é mostrar ao aluno um método simples para implementação de um teclado matricial.

20.2. Descrição

O teclado matricial do McMaster é composto por 4 linhas e 4 colunas formando assim uma matriz de 16 teclas. Desta forma, utilizam-se 8 pinos do microcontrolador para realizar a leitura do teclado.

Os conceitos adotados na experiência de varredura de displays de leds não são muito diferentes dos adotados para varrer os estados de um teclado matricial.

Analisando o esquema elétrico nota-se que todas as teclas de uma mesma coluna estão interligadas. Além disso, nota-se um resistor de pull-down em cada uma das vias. Veja também que todas as teclas de uma mesma linha também encontram-se interligadas. A idéia de varredura aplicada aqui é habilitar uma linha de cada vez e analisar se alguma tecla da linha habilitada está pressionada.

Para isso, deve-se configurar o microcontrolador com os pinos das linhas como saída e os pinos das colunas como entrada. Note que se todas as linhas estiverem em nível lógico 0, ou seja, se nenhuma linha estiver habilitada, ao ler o estado das colunas sempre será lido o valor 0, estando as teclas pressionadas ou não. Na verdade o microcontrolador estará lendo o estado dos resistores que no caso são de pull-down, ou seja, leitura em 0.

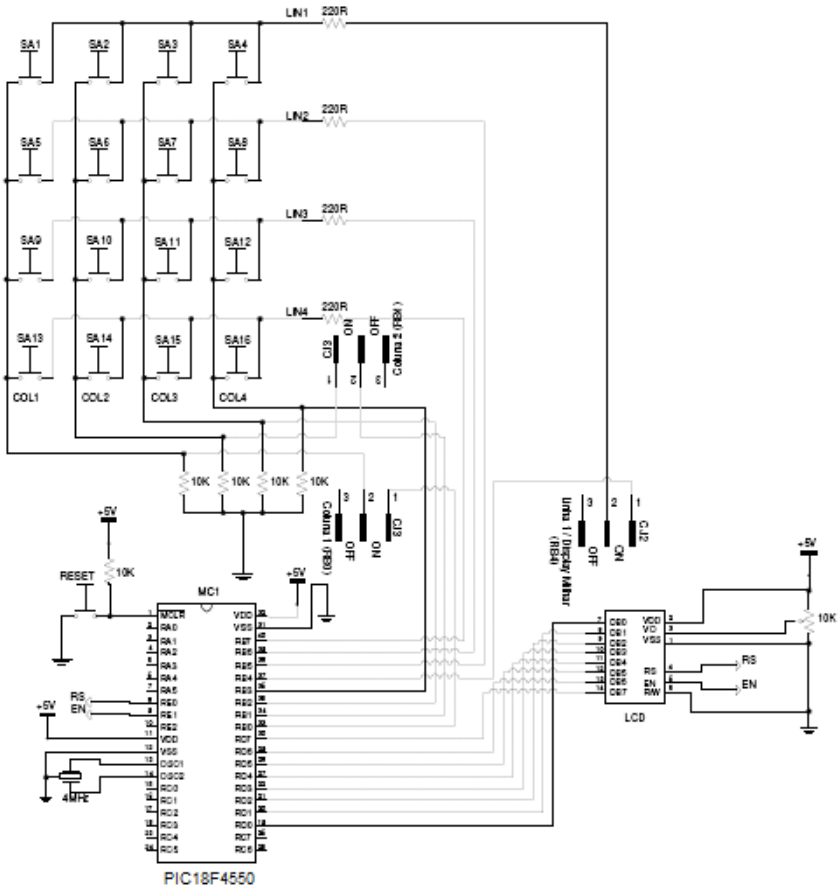
Porém, se habilitarmos uma das linhas (e apenas uma) colocando-a em nível lógico 1 e pressionarmos uma tecla dessa linha, ao lermos o estado das colunas encontraremos um bit em 1, sendo que a posição do bit em 1 sinalizará a coluna na qual a tecla foi pressionada. Como foi o próprio microcontrolador que habilitou a linha, o número da linha é conhecido e como a posição do bit em 1 define a coluna da tecla é fácil determinar a linha e coluna da tecla pressionada.

O conceito de varredura continua válido, pois apenas uma linha deve ser habilitada de cada vez e o microcontrolador deve ficar o tempo todo alterando (varrendo) a linha habilitada e lendo o estado das colunas. Enquanto nenhum bit das colunas valer 1, a varredura das linhas continua sendo executada. Ao encontrar uma coluna com tecla pressionada o software deve executar o filtro de debounce mantendo a linha atual habilitada até que o filtro de debounce seja finalizado.

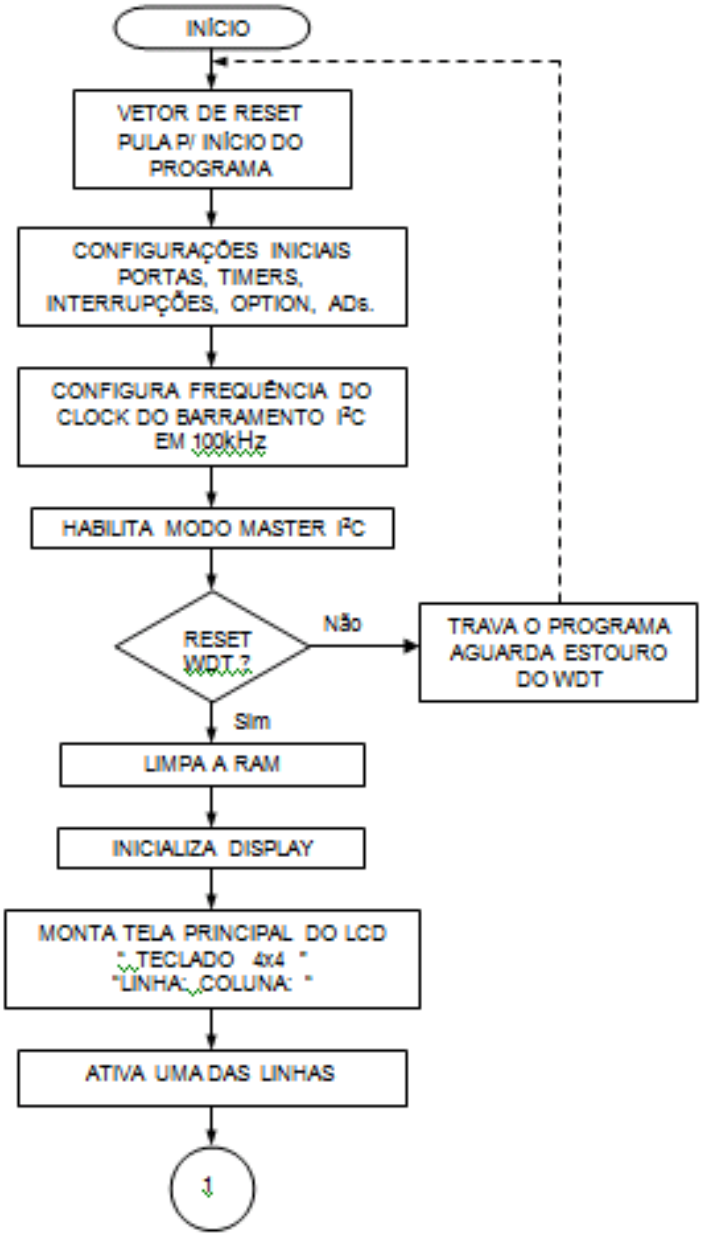
No fluxograma apresentado fica fácil de entender o conceito da varredura do teclado matricial.

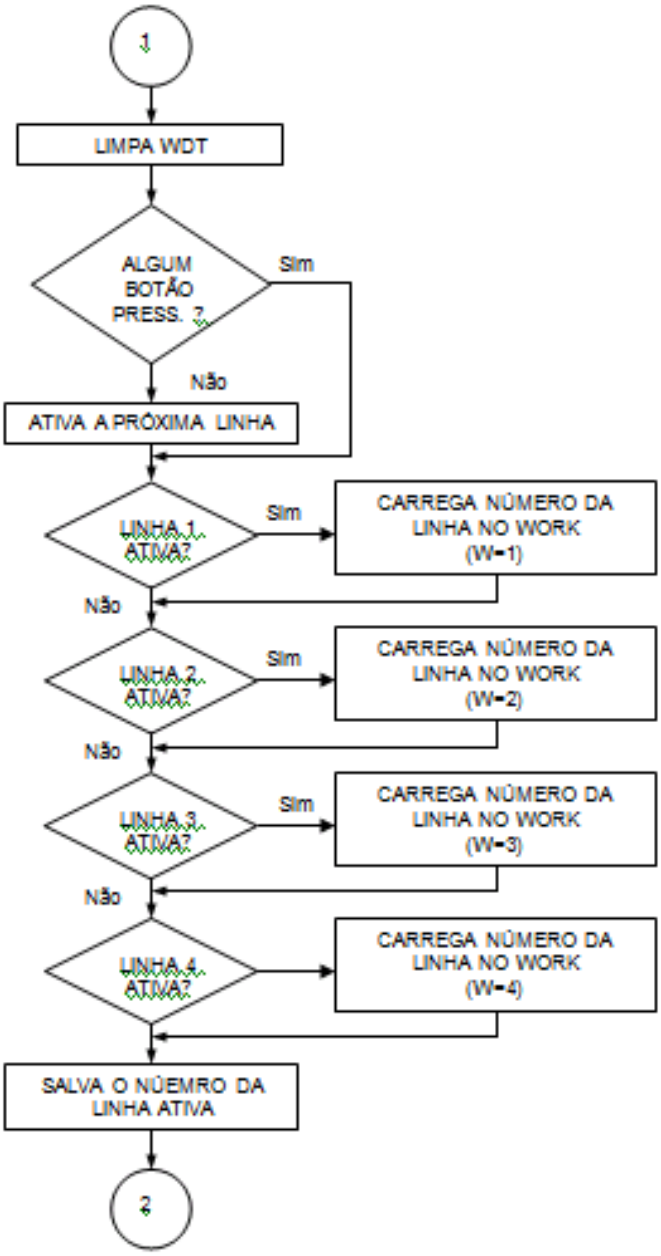
O exemplo da experiência analisa o teclado e caso alguma tecla seja pressionada, mostra a linha e coluna da mesma é mostrado no display LCD.

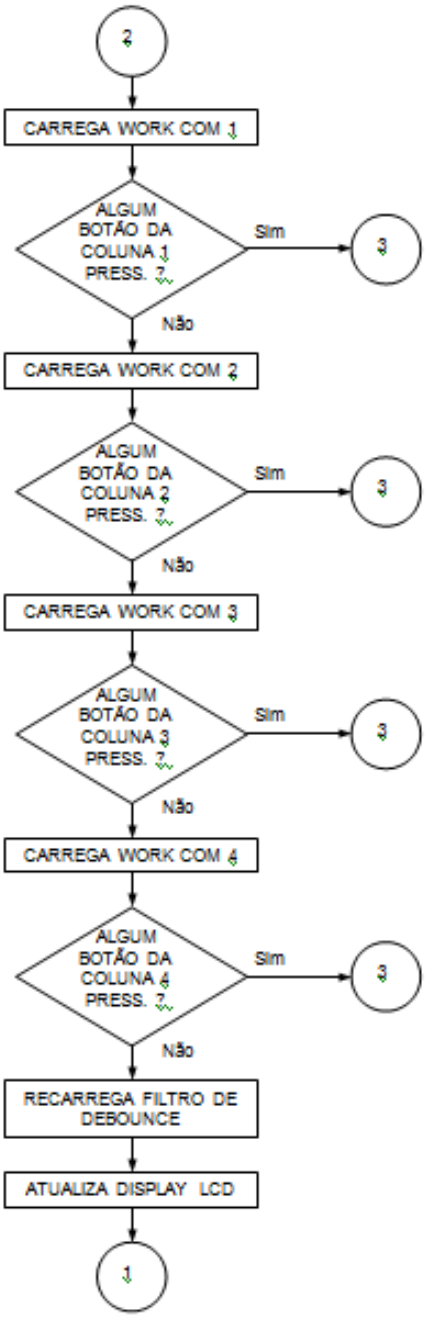
20.3. Esquema Elétrico

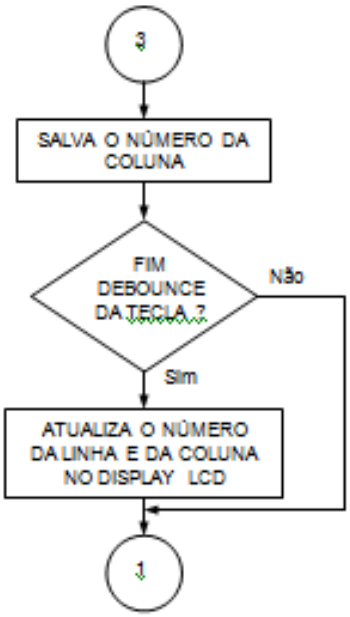


20.4. Fluxograma




1
LIMPA WDT
ALGUM BOTÃO PRESS. ?
Sim
Não
ATIVA A PRÓXIMA LINHA
LINHA 1 ATIVA?
Sim
CARREGA NÚMERO DA LINHA NO WORK (W=1)
Não
LINHA 2 ATIVA?
Sim
CARREGA NÚMERO DA LINHA NO WORK (W=2)
Não
LINHA 3 ATIVA?
Sim
CARREGA NÚMERO DA LINHA NO WORK (W=3)
Não
LINHA 4 ATIVA?
CARREGA NÚMERO DA LINHA NO WORK (W=4)
SALVA O NÚEMRO DA LINHA ATIVA
2


2
CARREGA WORK COM 1
ALGUM BOTÃO DA COLUNA 1 PRESS. ?
Sim
3
Não
CARREGA WORK COM 2
ALGUM BOTÃO DA COLUNA 2 PRESS. ?
Sim
3
Não
CARREGA WORK COM 3
ALGUM BOTÃO DA COLUNA 3 PRESS. ?
Sim
3
Não
CARREGA WORK COM 4
ALGUM BOTÃO DA COLUNA 4 PRESS. ?
Sim
3
Não
RECARREGA FILTRO DE DEBOUNCE
ATUALIZA DISPLAY LCD
1


3
SALVA O NÚMERO DA COLUNA
FIM DEBOUNCE DA TECLA ?
Não
Sim
ATUALIZA O NÚMERO DA LINHA E DA COLUNA NO DISPLAY LCD
1

20.5. Código

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Exemplo de Código para utilização com a McMaster            *
 *                                 Exemplo 18                               *
 *                                                                          *
 *                                  MOSAICO                                 *
 *                                                                          *
 *  TEL: (0XX11) 4992-8775                 SITE: www.mosaico.com.br         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                           *
 *   DATA : 18/01/2010                                                      *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                             Descrição geral                              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* Este exemplo foi elaborado para explicar o funcionamento do teclado
   matricial4x4.
   O número da linha e coluna da tecla pressionada e mostrado no LCD.*/

/*      CONFIGURAÇÂO DOS JUMPERS DA PLACA:

        CAP USB(RC3)                -> ON
        DATA I2C                    -> OFF
        CLOCK I2C                   -> OFF
        COLUNA 2(RB1)               -> ON
        COLUNA 1(RB0)               -> ON
        LINHA 1 / DISPLAY MILHAR(RB4)    -> ON
        LEDS ESPECIAS (RC0,RC1,RC2)      -> ON

*/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                             DEFINIÇÃO PIC                                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <p18F4550.h>           // Register definitions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               INCLUDES DAS FUNÇÕES DE PERIFÉRICOS DO PIC                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <pwm.h>                //PWM library functions
#include <adc.h>                //ADC library functions
#include <timers.h>             //Timer library functions
#include <delays.h>             //Delay library functions
#include <i2c.h>                //I2C library functions
#include <stdlib.h>             //Library functions
#include <usart.h>              //USART library functions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                         Configurações para gravação                      *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#pragma config FOSC = HS
#pragma config CPUDIV = OSC1_PLL2
#pragma config WDT = OFF
#pragma config WDTPS = 128
#pragma config LVP = OFF
#pragma config PWRT = ON
```

```
#pragma config BOR = OFF
#pragma config BORV = 0
#pragma config PBADEN = OFF
#pragma config DEBUG = OFF
#pragma config PLLDIV = 1
#pragma config USBDIV = 1
#pragma config FCMEN = OFF
#pragma config IESO = OFF
#pragma config VREGEN = OFF
#pragma config MCLRE = ON
#pragma config LPT1OSC = OFF
#pragma config CCP2MX = ON
#pragma config STVREN = OFF
#pragma config ICPRT = OFF

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Definição e inicialização das variáveis Globais           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Constantes internas                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de constantes facilita a programação e a manutenção.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Declaração dos flags de software                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de flags ajuda na programação e economiza memória RAM.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                 Definição e inicialização dos port's                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      PROTOTIPAGEM DE FUNÇÕES                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void comando_lcd(unsigned char caracter);
void escreve_lcd(unsigned char caracter);
void limpa_lcd(void);
void inicializa_lcd(void);
void tela_principal(void);
void trata_coluna(void);

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Constantes internas                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de constantes facilita a programação e a manutenção.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                Definição e inicialização das variáveis                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.

    unsigned char filtro;       // Filtro para teclas
    unsigned char num_linha;    // Armazena o número da linha ativada
    unsigned char num_coluna;   // Armazena o número da coluna
//    unsigned char linha;      // Registrador auxiliar para ativar as linhas do
teclado 4x4

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Declaração dos flags de software                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de flags ajuda na programação e economiza memória RAM.
union Linhas_teclado{
  struct{
```

```
    unsigned BIT0:1;
    unsigned BIT1:1;
    unsigned BIT2:1;
    unsigned BIT3:1;
    unsigned BIT4:1;
    unsigned BIT5:1;
    unsigned BIT6:1;
    unsigned BIT7:1;
  };                          //ARMAZENA OS FLAGS DE CONTROLE
    unsigned char linha_teclado;
}linha_ativa;

#define     linha1  linha_ativa.BIT4    /* LINHA_ATIVA,4, BIT 4 DO REGISTRADOR
LINHA_ATIVA
                                        REPRESENTA A LINHA 1 DO TECLADO 4x4
                                        1 -> LINHA ESTÁ ATIVADA
                                        0 -> LINHA ESTÁ DESATIVADA */

#define     linha2  linha_ativa.BIT5    /* LINHA_ATIVA,5, BIT 5 DO REGISTRADOR
LINHA_ATIVA
                                        REPRESENTA A LINHA 2 DO TECLADO 4x4
                                        1 -> LINHA ESTÁ ATIVADA
                                        0 -> LINHA ESTÁ DESATIVADA */

#define     linha3  linha_ativa.BIT6    /* LINHA_ATIVA,6, BIT 6 DO REGISTRADOR
LINHA_ATIVA
                                        REPRESENTA A LINHA 3 DO TECLADO 4x4
                                        1 -> LINHA ESTÁ ATIVADA
                                        0 -> LINHA ESTÁ DESATIVADA */

#define     linha4  linha_ativa.BIT7    /* LINHA_ATIVA,7, BIT 7 DO REGISTRADOR
LINHA_ATIVA
                                        REPRESENTA A LINHA 4 DO TECLADO 4x4
                                        1 -> LINHA ESTÁ ATIVADA
                                        0 -> LINHA ESTÁ DESATIVADA */

#define     fim     linha_ativa.BIT3    /* REPRESENTA O FINAL DA VARREDURA
                                        1 -> LINHA ESTÁ ATIVADA
                                        0 -> LINHA ESTÁ DESATIVADA */

#define     linha   linha_ativa.linha_teclado

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              ENTRADAS                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// As entradas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.
#define     coluna1         PORTBbits.0     /* PINO DE ENTRADA DA COLUNA 1
                                            1 -> ALGUMA TECLA DESTA COLUNA ESTÁ
PRESSIONADA
                                            0 -> NENHUMA TECLA DESTA COLUNA ESTÁ
PRESSIONADA */

#define     coluna2         PORTBbits.1     /* PINO DE ENTRADA DA COLUNA 2
                                            1 -> ALGUMA TECLA DESTA COLUNA ESTÁ
PRESSIONADA
                                            0 -> NENHUMA TECLA DESTA COLUNA ESTÁ
PRESSIONADA */

#define     coluna3         PORTBbits.2     /* PINO DE ENTRADA DA COLUNA 3
                                            1 -> ALGUMA TECLA DESTA COLUNA ESTÁ
PRESSIONADA
                                            0 -> NENHUMA TECLA DESTA COLUNA ESTÁ
PRESSIONADA */

#define     coluna4         PORTBbits.3     /* PINO DE ENTRADA DA COLUNA 4
```

```
                                                1 -> ALGUMA TECLA DESTA COLUNA ESTÁ
PRESSIONADA
                                                0 -> NENHUMA TECLA DESTA COLUNA ESTÁ
PRESSIONADA */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                                SAÍDAS                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// As saídas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define     rs      PORTEbits.RE0       // via do lcd que sinaliza recepção de dados ou
comando
#define     enable  PORTEbits.RE1       // enable do lcd

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Rotina que envia um COMANDO para o LCD                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void comando_lcd(unsigned char caracter)
{
    rs = 0;                     // seleciona o envio de um comando
    PORTD = caracter;           // carrega o PORTD com o caracter
    enable = 1 ;                // gera pulso no enable
    Delay10TCYx(1);             // espera 3 microsegundos
    enable = 0;                 // desce o pino de enable
    Delay10TCYx(4);             // espera mínimo 40 microsegundos
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               Rotina que envia um DADO a ser escrito no LCD             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void escreve_lcd(unsigned char caracter)
{
    rs = 1;                     // seleciona o envio de um comando
    PORTD = caracter;           // carrega o PORTD com o caracter
    enable = 1;                 // gera pulso no enable
    Delay10TCYx(1);             // espera 10 microsegundos
    enable = 0;                 // desce o pino de enable
    Delay10TCYx(4);             // espera mínimo 40 microsegundos
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                             Função para limpar o LCD                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void limpa_lcd(void)
{
    comando_lcd(0x01);          // limpa lcd
    Delay1KTCYx(2);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Inicialização do Display de LCD                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void inicializa_lcd(void)
{
    comando_lcd(0x30);          // envia comando para inicializar display
    Delay1KTCYx(4);             // espera 4 milisengundos

    comando_lcd(0x30);          // envia comando para inicializar display
    Delay10TCYx(10);            // espera 100 microsengundos

    comando_lcd(0x30);          // envia comando para inicializar display
```

```
    comando_lcd(0x38);          // liga o display, sem cursor e sem blink

    limpa_lcd();                // limpa lcd

    comando_lcd(0x0c);          // display sem cursor

    comando_lcd(0x06);          // desloca cursor para a direita
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Tela Principal                                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void tela_principal(void)
{
    comando_lcd(0x82);     // posiciona o cursor na linha 0, coluna 0
    escreve_lcd ('T');     // imprime mensagem no lcd
    escreve_lcd ('e');
    escreve_lcd ('c');
    escreve_lcd ('l');
    escreve_lcd ('a');
    escreve_lcd ('d');
    escreve_lcd ('o');
    escreve_lcd (' ');
    escreve_lcd ('4');
    escreve_lcd ('X');
    escreve_lcd ('4');

    comando_lcd(0xC0);     // posiciona o cursor na linha 1, coluna 0
    escreve_lcd ('L');     // imprime mensagem no lcd
    escreve_lcd ('i');
    escreve_lcd ('n');
    escreve_lcd ('h');
    escreve_lcd ('a');
    escreve_lcd (':');
    escreve_lcd (' ');
    escreve_lcd (' ');
    escreve_lcd ('C');
    escreve_lcd ('o');
    escreve_lcd ('l');
    escreve_lcd ('u');
    escreve_lcd ('n');
    escreve_lcd ('a');
    escreve_lcd (':');
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *            Rotina para tratamento das colunas do Teclado 4X4            *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void trata_coluna(void)
{
    if(filtro)
        {
        filtro--;
        if (filtro)return;
        }
        PORTB = 0;
        comando_lcd(0xC6);              // posiciona o cursor na linha 1, coluna 6
        escreve_lcd (0x30 + num_linha); //
        comando_lcd(0xCF);              // posiciona o cursor na linha 1, coluna 15
        escreve_lcd (0x30 + num_coluna);//
        PORTD = 0;
        PORTB = linha;
}
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Configurações do Pic                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void main()
{

    PORTA = 0x00;               //Limpa PORTA
    PORTB = 0x00;               //Limpa PORTB
    PORTC = 0x00;               //Limpa PORTC
    PORTD = 0x00;               //Limpa PORTD
    PORTE = 0x00;               //Limpa PORTE

    LATA = 0x00;                //Limpa PORTA
    LATB = 0x00;                //Limpa PORTB
    LATC = 0x00;                //Limpa PORTC
    LATD = 0x00;                //Limpa PORTD
    LATE = 0x00;                //Limpa PORTE

    TRISA=(0b00101111);        // configuração da direção dos pinos de I/O
    TRISB=(0b00001111);
    TRISC=(0b10011000);
    TRISD=(0b00000000);
    TRISE=(0b00000000);

    ADCON1=0b00001111;

    inicializa_lcd();           // configura o lcd
    tela_principal();           // imprime a tela principal no LCD

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Rotina principal                                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
    linha = 0b00010000;

    while(1)                    // rotina principal
    {
        ClrWdt();               //Inicia o watch-dog timer

ativa_proxima_linha:
        if (filtro == 100)
        {
            linha = linha >> 1;
            if (fim) linha = 0b10000000;
            PORTB = linha;
        }
        if (linha4) num_linha = 4;     // verifica qual é a linha ativa
        if (linha3) num_linha = 3;
        if (linha2) num_linha = 2;
        if (linha1) num_linha = 1;

        switch(PORTB & 0b00001111)
        {
            case 1:
                num_coluna = 1;        // verifica qual é a linha ativa
                trata_coluna();
            break;

            case 2:
                num_coluna = 2;        // verifica qual é a linha ativa
                trata_coluna();
            break;

            case 4:
```

```
                num_coluna = 3;         // verifica qual é a linha ativa
                trata_coluna();
            break;

            case 8:
                num_coluna = 4;         // verifica qual é a linha ativa
                trata_coluna();
            break;

            default:
                filtro = 100;
                PORTB = 0;              // Desativa todos os displays de 7 segmentos
                comando_lcd(0xC6);      // posiciona o cursor na linha 1, coluna 6
                escreve_lcd (' ');      //
                comando_lcd(0xCF);      // posiciona o cursor na linha 1, coluna 15
                escreve_lcd (' ');      //
                PORTD = 0;
                PORTB = linha;
        }
    }
}
```

20.6. Dicas e Comentários

Notar que ao habilitar uma linha do teclado matricial também se habilita um dos displays de 7 segmentos, ou seja, foram utilizados os mesmos pinos do microcontrolador para habilitar as linhas e os displays. Esta é uma forma econômica de varrer os displays e ao mesmo tempo varrer o teclado. Veja que com apenas 16 I/Os do microcontrolador foi possível ligar 4 displays de 7 segmentos e 16 teclas. Foram utilizados 8 I/Os para segmentos mais o ponto decimal, 4 I/Os de seleção (linhas e displays) e 4 I/Os de colunas.

20.7. Exercícios Propostos

1. Alterar o mapeamento do software invertendo a numeração das linhas e colunas.
2. Mostrar no display LCD o número da tecla pressionada e não o número da linha e da coluna.
3. Sem utilizar o display LCD, fazer um software que utilize a varredura do teclado e dos displays ao mesmo tempo, mostrando o número da tecla pressionada nos displays.

21. Experiência 19 – Relógio de tempo real (RTC)

21.1. Objetivo

O objetivo desta experiência é mostrar como utilizar o relógio de tempo real (RTC).

21.2. Descrição

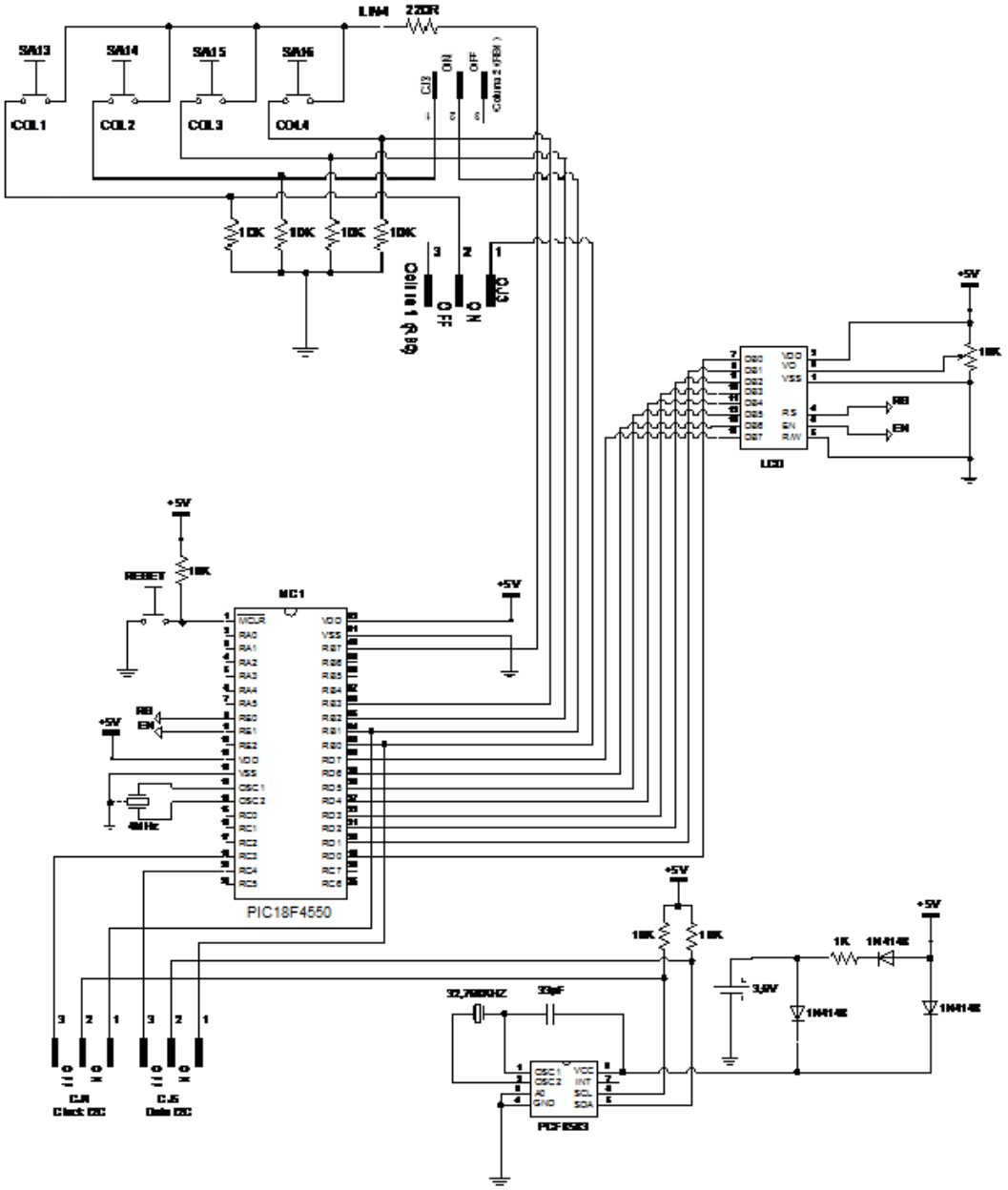
No McMaster optou-se pela utilização do relógio de tempo real (RTC) modelo PCF8583P da Philips. O protocolo de comunicação é do tipo I^2C e para maiores informações a respeito do componente deve-se consultar o data sheet disponível no CD.

As rotinas de acesso ao relógio seguem o padrão adotado na experiência 16 (Master I^2C) com apenas algumas modificações. Na realidade as alterações são que o relógio está mapeado no endereço 0h (000b) para evitar conflitos com a memória e ao invés de serem utilizados dois bytes para compor o endereço foi utilizado apenas um o que já é suficiente.

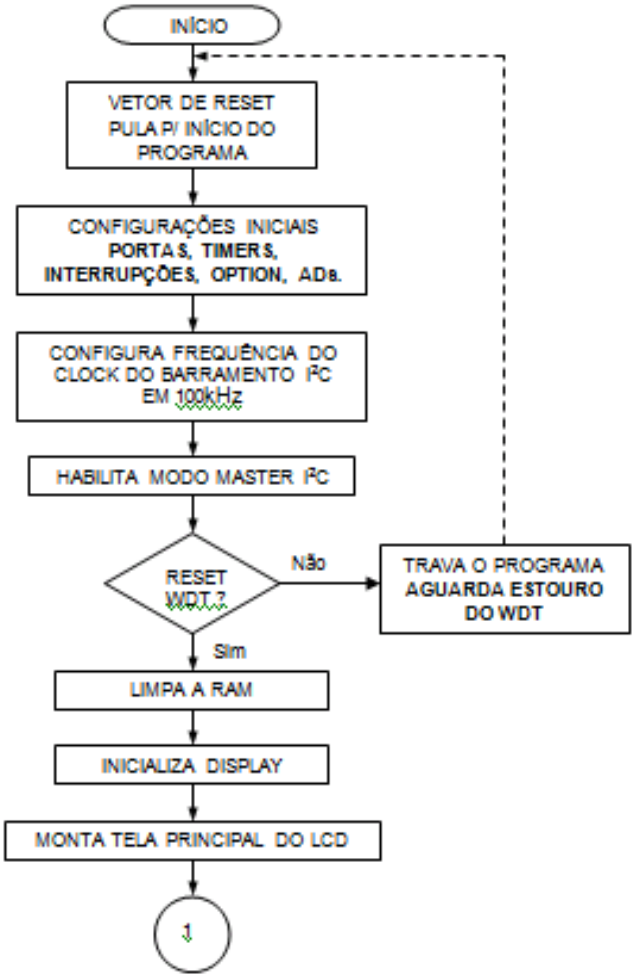
O software da experiência apenas lê a hora atual do relógio e mostra o resultado no LCD.

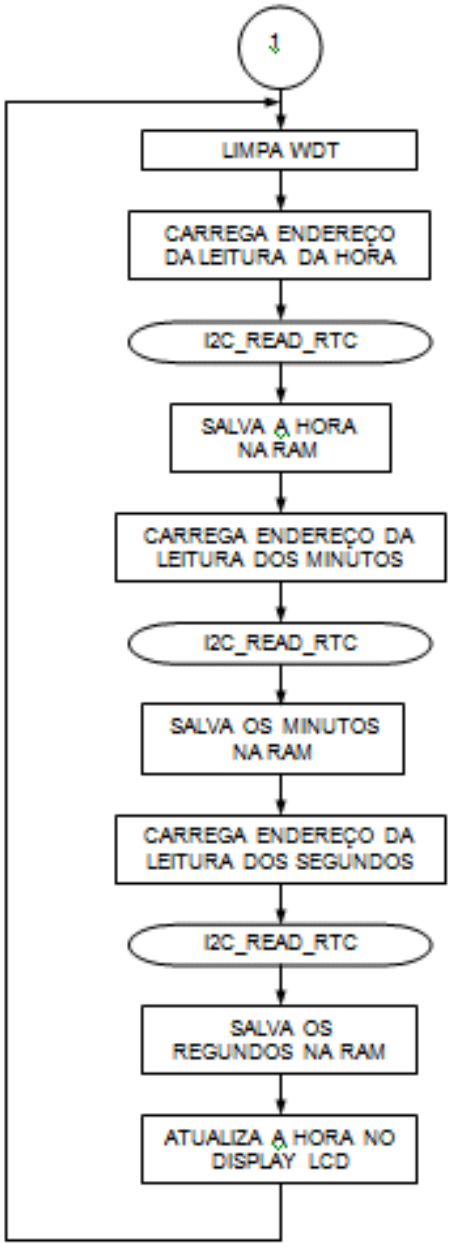
Vale lembrar que o relógio é completo, ou seja, dispõe de hora, minuto, segundo, dia, mês e ano, inclusive bissexto.

21.3. Esquema Elétrico

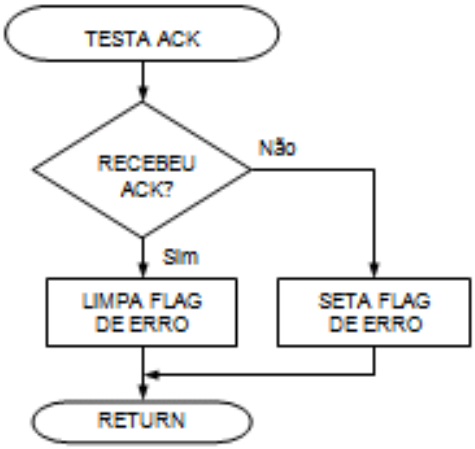


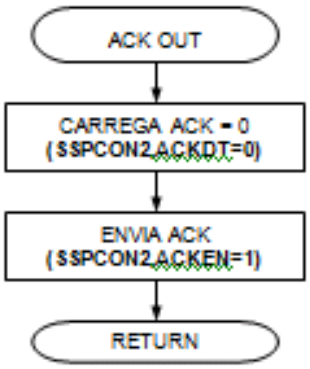
21.4. Fluxograma

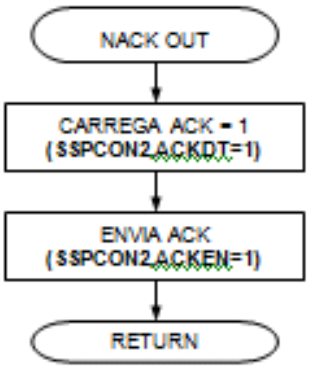

INÍCIO
VETOR DE RESET PULA P/ INÍCIO DO PROGRAMA
CONFIGURAÇÕES INICIAIS PORTAS, TIMERS, INTERRUPÇÕES, OPTION, ADs.
CONFIGURA FREQUÊNCIA DO CLOCK DO BARRAMENTO I²C EM 100kHz
HABILITA MODO MASTER I²C
RESET WDT ?
Não
TRAVA O PROGRAMA AGUARDA ESTOURO DO WDT
Sim
LIMPA A RAM
INICIALIZA DISPLAY
MONTA TELA PRINCIPAL DO LCD
1

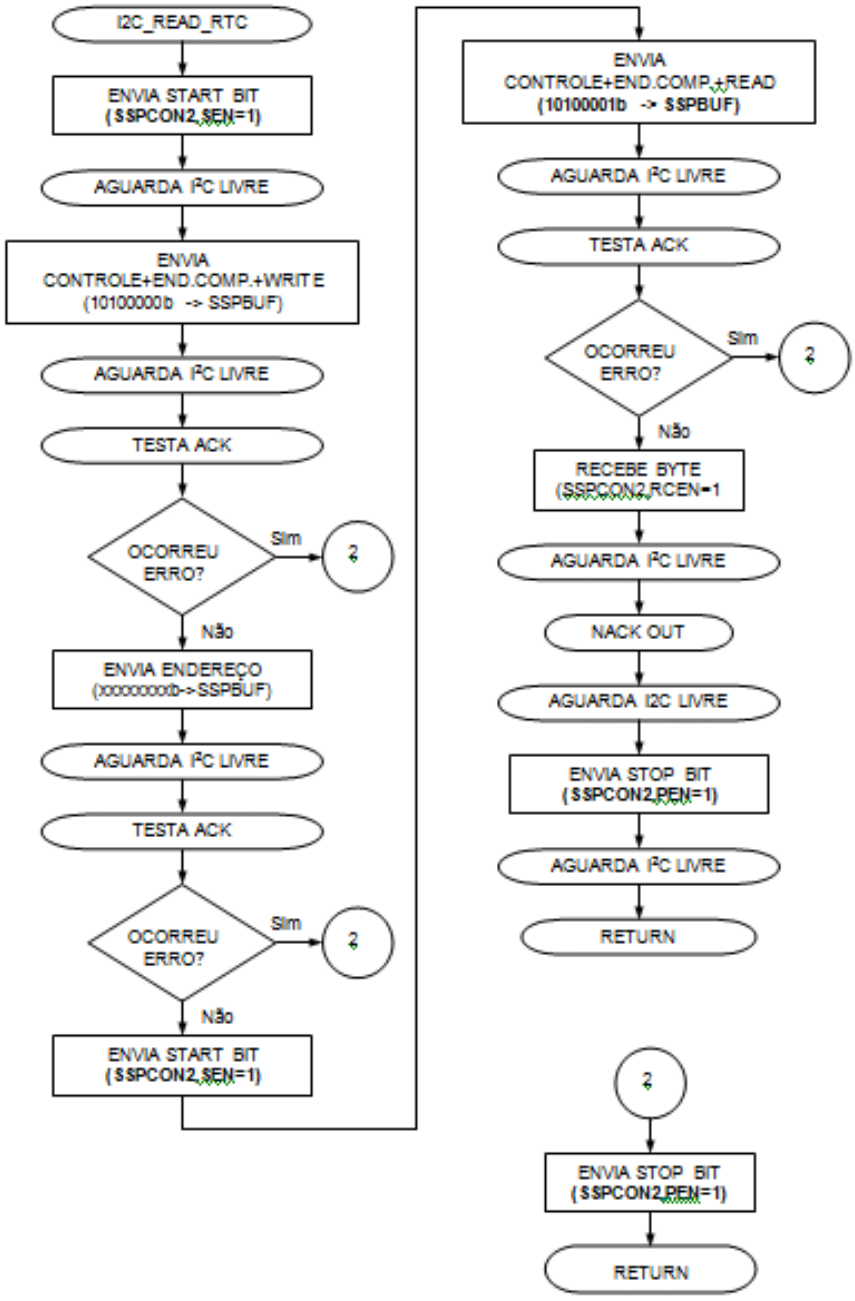

1
LIMPA WDT
CARREGA ENDEREÇO DA LEITURA DA HORA
I2C_READ_RTC
SALVA A HORA NA RAM
CARREGA ENDEREÇO DA LEITURA DOS MINUTOS
I2C_READ_RTC
SALVA OS MINUTOS NA RAM
CARREGA ENDEREÇO DA LEITURA DOS SEGUNDOS
I2C_READ_RTC
SALVA OS REGUNDOS NA RAM
ATUALIZA A HORA NO DISPLAY LCD

AGUARDA I²C LIVRE
EVENTO I²C EM ANDAMENTO?
Sim
Não
TODOS OS EVENTO I²C FINALIZADOS?
Não
RETURN


TESTA ACK
RECEBEU ACK?
Não
Sim
LIMPA FLAG DE ERRO
SETA FLAG DE ERRO
RETURN


ACK OUT
CARREGA ACK = 0 (SSPCON2.ACKDT=0)
ENVIA ACK (SSPCON2.ACKEN=1)
RETURN


NACK OUT
CARREGA ACK = 1 (SSPCON2.ACKDT=1)
ENVIA ACK (SSPCON2.ACKEN=1)
RETURN


I2C_READ_RTC
ENVIA START BIT (SSPCON2.SEN=1)
AGUARDA I²C LIVRE
ENVIA CONTROLE+END.COMP.+WRITE (10100000b -> SSPBUF)
AGUARDA I²C LIVRE
TESTA ACK
OCORREU ERRO?
Sim
2
Não
ENVIA ENDEREÇO (xxxxxxxxb->SSPBUF)
AGUARDA I²C LIVRE
TESTA ACK
OCORREU ERRO?
Sim
2
Não
ENVIA START BIT (SSPCON2.SEN=1)
ENVIA CONTROLE+END.COMP.+READ (10100001b -> SSPBUF)
AGUARDA I²C LIVRE
TESTA ACK
OCORREU ERRO?
Sim
2
Não
RECEBE BYTE (SSPCON2.RCEN=1
AGUARDA I²C LIVRE
NACK OUT
AGUARDA I2C LIVRE
ENVIA STOP BIT (SSPCON2.PEN=1)
AGUARDA I²C LIVRE
RETURN
2
ENVIA STOP BIT (SSPCON2.PEN=1)
RETURN

21.5. Código

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Exemplo de Código para utilização com a McMaster            *
 *                                Exemplo 19                               *
 *                                                                         *
 *                                 MOSAICO                                 *
 *                                                                         *
 *  TEL: (0XX11) 4992-8775                SITE: www.mosaico.com.br         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                          *
 *   DATA : 18/01/2010                                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            Descrição geral                              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Exemplo de utilização do relógio de tempo real PCF8583 através da interface
//I2C. Para maiores informações a respeito do integrado consultar o datasheet
//do componente.

/*      CONFIGURAÇÂO DOS JUMPERS DA PLACA:

        CAP USB(RC3)                -> ON
        DATA I2C                    -> ON
        CLOCK I2C                   -> ON
        COLUNA 2(RB1)               -> OFF
        COLUNA 1(RB0)               -> OFF
        LINHA 1 / DISPLAY MILHAR(RB4)    -> ON
        LEDS ESPECIAS (RC0,RC1,RC2)      -> ON

*/


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           DEFINIÇÃO PIC                                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <p18F4550.h>             // Register definitions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               INCLUDES DAS FUNÇÕES DE PERIFÉRICOS DO PIC                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <pwm.h>                //PWM library functions
#include <adc.h>                //ADC library functions
#include <timers.h>             //Timer library functions
#include <delays.h>             //Delay library functions
#include <i2c.h>                //I2C library functions
#include <stdlib.h>             //Library functions
#include <usart.h>              //USART library functions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      Configurações para gravação                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#pragma config FOSC = HS
#pragma config CPUDIV = OSC1_PLL2
#pragma config WDT = OFF
#pragma config WDTPS = 128
```

```
#pragma config LVP = OFF
#pragma config PWRT = ON
#pragma config BOR = OFF
#pragma config BORV = 0
#pragma config PBADEN       = OFF
#pragma config DEBUG        = OFF
#pragma config PLLDIV       = 1
#pragma config USBDIV       = 1
#pragma config FCMEN        = OFF
#pragma config IESO  = OFF
#pragma config VREGEN       = OFF
#pragma config MCLRE        = ON
#pragma config LPT1OSC      = OFF
#pragma config CCP2MX = ON
#pragma config STVREN       = OFF
#pragma config ICPRT        = OFF

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Definição e inicialização das variáveis Globais          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Constantes internas                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de constantes facilita a programação e a manutenção.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Declaração dos flags de software                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de flags ajuda na programação e economiza memória RAM.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Definição e inicialização dos port's                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       PROTOTIPAGEM DE FUNÇÕES                         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void comando_lcd(unsigned char caracter);
void escreve_lcd(unsigned char caracter);
void limpa_lcd(void);
void inicializa_lcd(void);
void tela_principal(void);
void LE_RELOGIO(unsigned char endereco);
void ESCREVE_RELOGIO(unsigned char endereco);

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Constantes internas                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de constantes facilita a programação e a manutenção.
#define     END_HORA        4       // ENDEREÇO DA HORA NO RELOGIO RTC

#define     END_MINUTO      3       // ENDEREÇO DO MINUTO NO RELOGIO RTC

#define     END_SEGUNDO     2       // ENDEREÇO DO SEGUNDO NO RELOGIO RTC

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                Definição e inicialização das variáveis               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.
unsigned char dez_hora;
unsigned char uni_hora;
unsigned char dez_min;
unsigned char uni_min;
unsigned char dez_seg;
```

```
unsigned char uni_seg;
unsigned char buffer;

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Declaração dos flags de software                      *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de flags ajuda na programação e economiza memória RAM.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                               ENTRADAS                                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// As entradas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                                SAÍDAS                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// As saídas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define     rs      PORTEbits.RE0       // via do lcd que sinaliza recepção de dados ou
comando
#define     enable  PORTEbits.RE1       // enable do lcd

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Rotina que envia um COMANDO para o LCD                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void comando_lcd(unsigned char caracter)
{
    rs = 0;                     // seleciona o envio de um comando
    PORTD = caracter;           // carrega o PORTD com o caracter
    enable = 1 ;                // gera pulso no enable
    Delay10TCYx(1);             // espera 3 microsegundos
    enable = 0;                 // desce o pino de enable
    Delay10TCYx(4);             // espera mínimo 40 microsegundos
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               Rotina que envia um DADO a ser escrito no LCD             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void escreve_lcd(unsigned char caracter)
{
    rs = 1;                     // seleciona o envio de um comando
    PORTD = caracter;           // carrega o PORTD com o caracter
    enable = 1;                 // gera pulso no enable
    Delay10TCYx(1);             // espera 10 microsegundos
    enable = 0;                 // desce o pino de enable
    Delay10TCYx(4);             // espera mínimo 40 microsegundos
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            Função para limpar o LCD                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void limpa_lcd(void)
{
    comando_lcd(0x01);          // limpa lcd
    Delay1KTCYx(2);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Inicialização do Display de LCD                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
```

```
void inicializa_lcd(void)
{
    comando_lcd(0x30);          // envia comando para inicializar display
    Delay1KTCYx(4);             // espera 4 milisengundos

    comando_lcd(0x30);          // envia comando para inicializar display
    Delay10TCYx(10);            // espera 100 microsengundos

    comando_lcd(0x30);          // envia comando para inicializar display

    comando_lcd(0x38);          // liga o display, sem cursor e sem blink

    limpa_lcd();                // limpa lcd

    comando_lcd(0x0c);          // display sem cursor

    comando_lcd(0x06);          // desloca cursor para a direita
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          LEITURA DO RELÓGIO                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void LE_RELOGIO(unsigned char endereco)
{
    StartI2C();
    IdleI2C();              // espera fim do evento I2C
    WriteI2C(0b10100000);   // controle de escrita
    IdleI2C();              // espera fim do evento I2C
    WriteI2C(endereco);     // lê o relógio
    IdleI2C();              // espera fim do evento I2C
    RestartI2C();           // restart
    IdleI2C();              // espera fim do evento I2C
    WriteI2C(0b10100001);   // controle de leitura
    IdleI2C();              // espera fim do evento I2C
    buffer = ReadI2C();     // recebe dado do relógio
    IdleI2C();              // espera fim do evento I2C
    StopI2C();              // fim de comunicação
    IdleI2C();              // espera fim do evento I2C
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          ESCRITA NO RELÓGIO                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void ESCREVE_RELOGIO(unsigned char endereco)
{
    StartI2C();
    IdleI2C();              // espera fim do evento I2C
    WriteI2C(0b10100000);   // controle de escrita
    IdleI2C();              // espera fim do evento I2C
    WriteI2C(endereco);     // escreve dado no relógio
    IdleI2C();              // espera fim do evento I2C
    WriteI2C(buffer);       // escreve dado no relógio
    IdleI2C();              // espera fim do evento I2C
    StopI2C();
    IdleI2C();              // espera fim do evento I2C
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Tela Principal                              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void tela_principal(void)
{
    comando_lcd(0x83);    // posiciona o cursor na linha 0, coluna 0
    escreve_lcd ('R');
    escreve_lcd ('e');
```

```
    escreve_lcd ('l');
    escreve_lcd ('o');
    escreve_lcd ('g');
    escreve_lcd ('i');
    escreve_lcd ('o');    // imprime mensagem no lcd
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Configurações do Pic                             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void main()
{

    PORTA = 0x00;               //Limpa PORTA
    PORTB = 0x00;               //Limpa PORTB
    PORTC = 0x00;               //Limpa PORTC
    PORTD = 0x00;               //Limpa PORTD
    PORTE = 0x00;               //Limpa PORTE

    LATA = 0x00;                //Limpa PORTA
    LATB = 0x00;                //Limpa PORTB
    LATC = 0x00;                //Limpa PORTC
    LATD = 0x00;                //Limpa PORTD
    LATE = 0x00;                //Limpa PORTE

    TRISA=(0b11111111);        // configuração da direção dos pinos de I/O
    TRISB=(0b11111111);
    TRISC=(0b11110111);
    TRISD=(0b00000000);
    TRISE=(0b00000100);

    ADCON1=0b00001111;

    OpenI2C(MASTER,SLEW_OFF);
    SSPADD = 9;

    buffer = 0x23;          // acerta hora
    ESCREVE_RELOGIO(END_HORA);

    buffer = 0x40;          // acerta minuto
    ESCREVE_RELOGIO(END_MINUTO);

    buffer = 0x00;          // acerta segundos
    ESCREVE_RELOGIO(END_SEGUNDO);

    inicializa_lcd();       // configura o lcd
    tela_principal();       // imprime a tela principal no LCD

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            Rotina principal                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
    while(1)                // rotina principal
    {
        ClrWdt();           //Inicia o watch-dog timer
        LE_RELOGIO(END_HORA);
        dez_hora = buffer >> 4;
        dez_hora = (dez_hora & 0b00000011) + 0x30;
        uni_hora = buffer;
        uni_hora = (uni_hora & 0b00001111) + 0x30;

        LE_RELOGIO(END_MINUTO);
        dez_min = buffer >> 4;
        dez_min = (dez_min & 0b00001111) + 0x30;
        uni_min = buffer;
        uni_min = (uni_min & 0b00001111) + 0x30;
```

```
        LE_RELOGIO(END_SEGUNDO);
        dez_seg = buffer >> 4;
        dez_seg = (dez_seg & 0b00001111) + 0x30;
        uni_seg = buffer;
        uni_seg = (uni_seg & 0b00001111) + 0x30;

        comando_lcd(0xC3);      // posiciona o cursor na linha 0, coluna 3
        escreve_lcd (dez_hora); // DEZ. HORA
        escreve_lcd (uni_hora); // UNID. HORA
        escreve_lcd (':');      //
        escreve_lcd (dez_min);  // DEZ. MINUTO
        escreve_lcd (uni_min);  // UNID. MINUTO
        escreve_lcd (':');      //
        escreve_lcd (dez_seg);  // DEZ. SEGUNDO
        escreve_lcd (uni_seg);  // UNID. SEGUNDO
    }
}
```

21.6. Dicas e Comentários

Além do relógio RTC o componente PCF8583 da Philips apresenta também uma região da memória que pode ser utilizada com uma RAM convencional, porém com acesso via I^2C. Como geralmente o relógio sempre é utilizado junto com uma bateria, como no McMaster, este recurso da RAM pode ser útil para armazenar informações enquanto o microcontrolador não está energizado.

21.7. Exercícios Propostos

1. Altere o software para mostrar a data e a hora no display LCD.
2. Crie um software para ajustar a data e hora do relógio utilizando o teclado matricial.

22. Experiência 20 – Sistema de temperatura e tacômetro

22.1. Objetivo

O objetivo desta experiência é mostrar a maioria os recursos disponíveis na placa de experiências EXP01.

22.2. Descrição

O monitoramento da temperatura é um dos problemas mais clássicos enfrentado pela maioria dos projetistas. Como o McMaster possui um sensor de temperatura (diodo) e dois atuadores: aquecimento (resistência) e resfriamento (ventilador), nada melhor que implementar um sistema capaz de obter a temperatura atual para mostrá-la no LCD.

22.2.1. O sensor de temperatura

Para obter a temperatura ambiente, fez-se uso de um diodo. Como o diodo é um componente que apresenta uma queda de tensão sobre ele proporcional a temperatura do mesmo, basta monitorar a tensão para encontrar a temperatura.

Para isso, o circuito eletrônico faz uso de um diodo de sinal convencional (1N4148) ligado a um amplificador e a uma porta analógica do PIC. Também está ligado ao amplificador um potenciômetro a fim de alterar o off-set da curva, ajustando assim a temperatura com uma referência externa.

Internamente, o sistema trabalha com uma conversão A/D de 8 bits, gerando 256 possíveis valores de tensão para o diodo. Para cada valor obtido, tem-se uma temperatura relacionada. A rotina TABELA_TEMP, que se encontra no final do código apresentado neste capítulo, efetua a conversão entre a tensão lida (unidades de A/D) e a temperatura real. Nada mais é que uma tabela de conversão/linearização.

Esta tabela foi construída com base na curva de resposta do diodo utilizado em função da temperatura. Caso seja construído um sensor de temperatura com outro tipo de componente que gere uma tensão variável, basta refazer a tabela de conversão.

Uma vez convertida, a temperatura é então mostrada no LCD, na unidade de Celsius [°C] (lado direito).

22.2.2. O aquecimento

O software possibilita que o usuário aumente a temperatura sobre o diodo através do controle manual da resistência existente na placa. Isso é feito por intermédio de um dos PWMs do PIC, que se encontra ligado ao resistor.

Através dos botões das colunas 1 e 2 pode-se aumentar e diminuir o duty cycle do PWM, variando de 0 a 100%. Mantendo-se os botões pressionados o incremento/decremento é automático. O valor atual para o aquecimento é mostrado no LCD (lado esquerdo).

22.2.3. O resfriamento

Inversamente ao aquecimento, o software possibilita também o resfriamento do sistema através do ventilador, que é controlado pelo outro canal de PWM do PIC. Obviamente o sistema só é capaz de obter temperaturas levemente abaixo do valor ambiente, mas a intenção é poder criar variáveis diferentes de aquecimento e resfriamento.

Controla-se o ventilador pelo duty cycle do PWM, mas para o sistema ficar um pouco mais completo, mostra-se no LCD (centro) o valor da rotação do motor, em RPS (rotações por segundo). Isto é feito através do sensor óptico que se encontra instalado na base das hélices do ventilador. Os botões das colunas 3 e 4 são usados para aumentar e diminuir o duty cycle do PWM, variando de 0 a 100%. Mantendo-se os botões pressionados o incremento/decremento é automático.

Cada vez que uma das pás da hélice passa em frente ao sensor óptico, um pulso é transmitido ao PIC. Como este sinal está ligado ao pino RC1, utilizou-se o TMR1 com incremento externo para contabilizar a quantidade de pulsos gerados. A cada segundo (base de tempo gerada pela interrupção de TMR2), o total de pulsos é transferido para a variável CONT_VENT. Antes de mostrar o valor correto no LCD, deve-se dividir o total de pulsos durante 1 segundo (CONT_VENT) pelo número de paletas (pulsos por volta). Neste caso CONT_VENT é dividido por 7.

22.2.4. Comunicação serial

O sistema possibilita ainda que maiores implementações sejam feitas com o tratamento externo da temperatura, pois os valores obtidos são enviados automaticamente pela porta serial, a cada segundo. Com isso, é possível criar um pequeno programa de computador capaz de plotar esta temperatura no decorrer do tempo. Os dados também podem ser armazenados para cálculos ou consultas futuras.

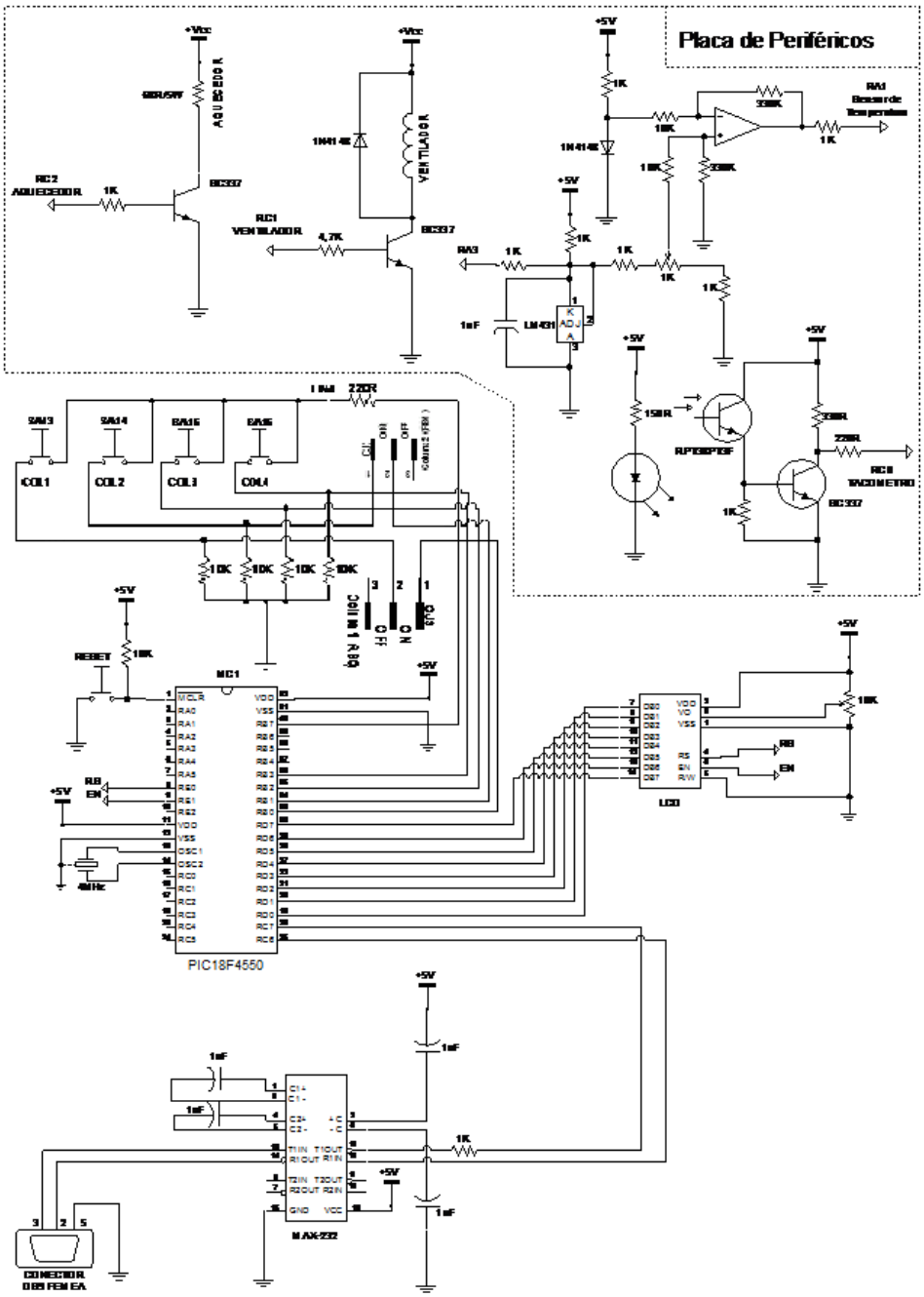
Os dados são transmitidos pela USART através do conector DB-9, respeitando-se o padrão RS-232 com 8N1 e baud rate de 9.600bps. A cada 1 segundo é transmitido um byte com o valor da temperatura já convertido para Celsius [°C].

22.2.5. Considerações gerais

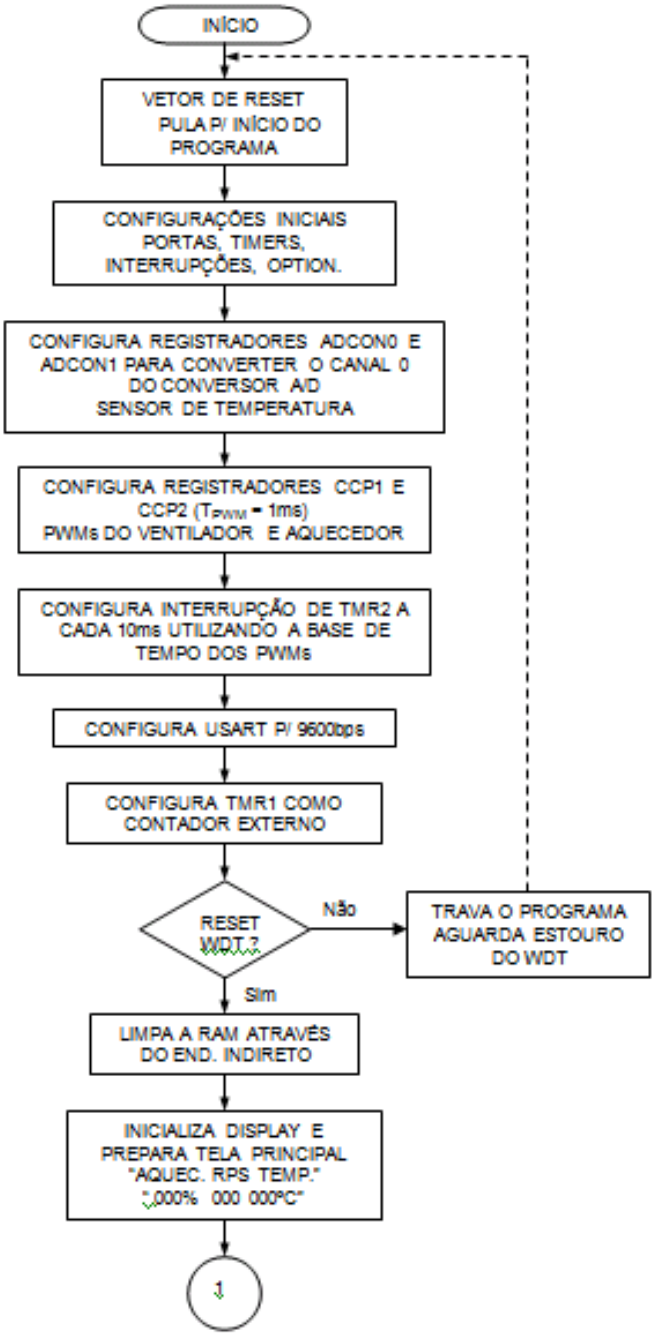
Ambos os PWMs são controlados pelo TMR2, que está regulado para 1ms. Por isso, o período dos 2 PWMs é de 1ms, ou seja, eles operam a 1kHz.

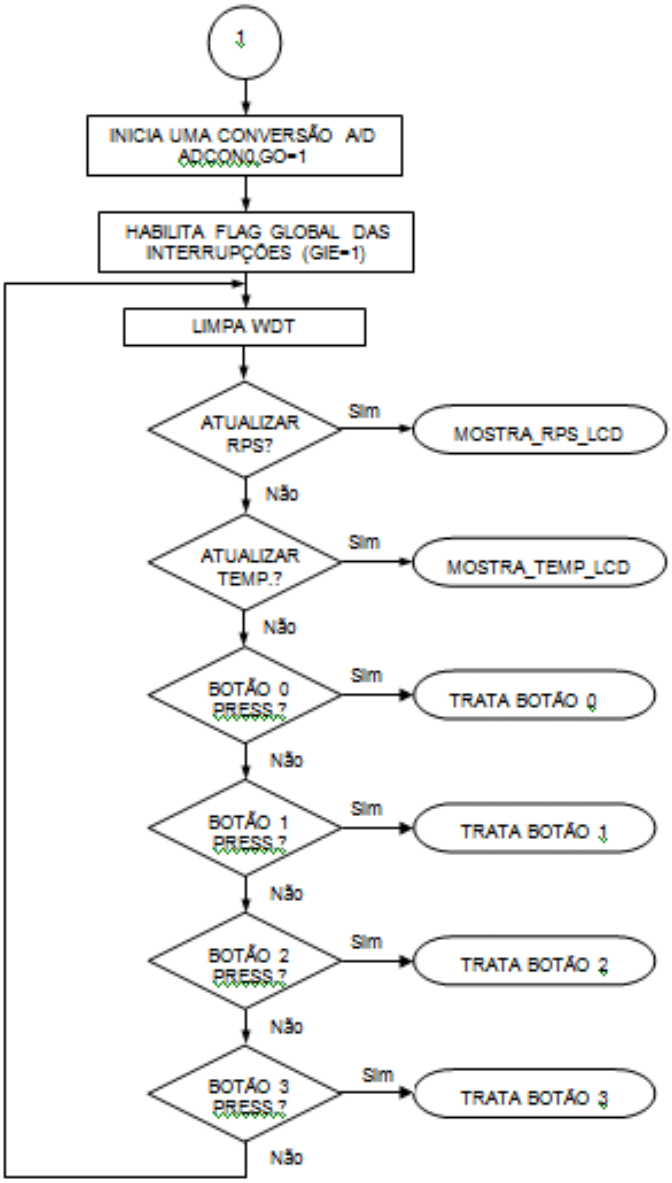
O postscale do TMR2 foi regulado em 1:10, gerando uma interrupção a cada 10ms. Utilizou-se um contador auxiliar (TEMPO_1S) para contabilizar 100 interrupções, gerando a base de tempo de 1 segundo. Esta base é utilizada para capturar a rotação do ventilador, efetuar uma conversão de temperatura e transmitir dados pela USART.

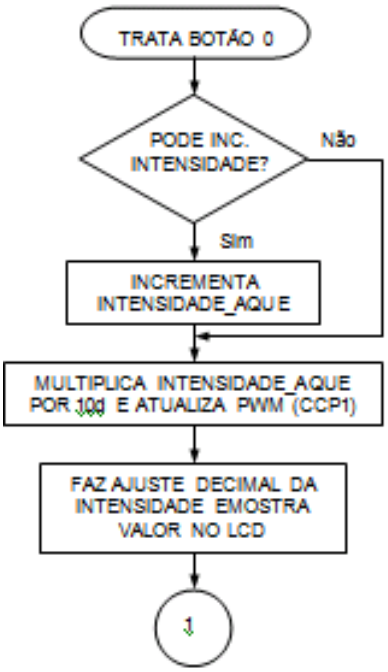
22.3. Esquema Elétrico

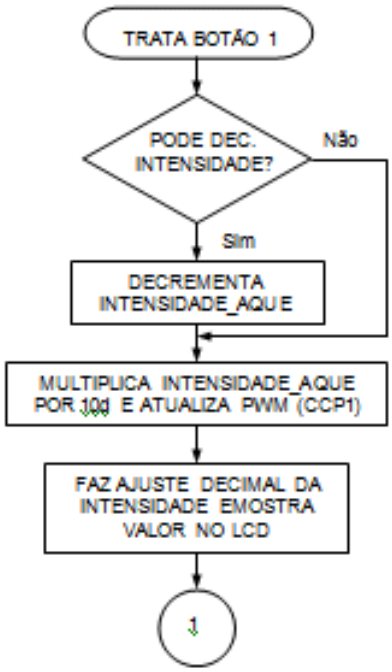


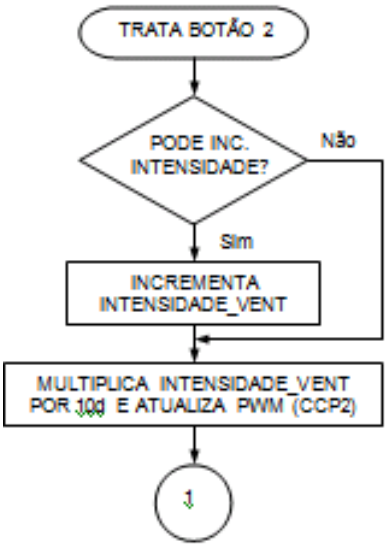
22.4. Fluxograma

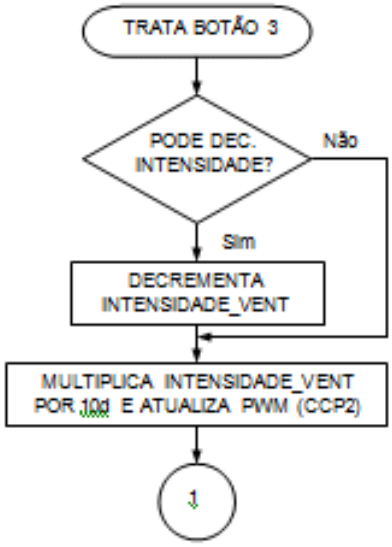


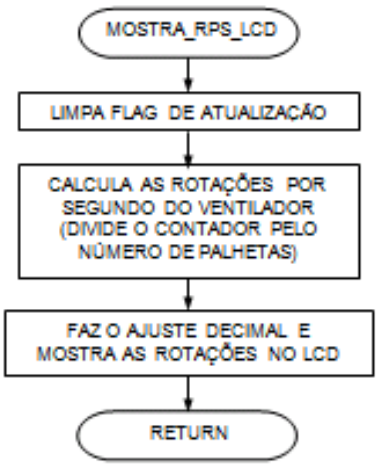

1
INICIA UMA CONVERSÃO A/D
ADCON0.GO=1
HABILITA FLAG GLOBAL DAS INTERRUPÇÕES (GIE=1)
LIMPA WDT
ATUALIZAR RPS?
Sim
MOSTRA_RPS_LCD
Não
ATUALIZAR TEMP.?
Sim
MOSTRA_TEMP_LCD
Não
BOTÃO 0 PRESS.?
Sim
TRATA BOTÃO 0
Não
BOTÃO 1 PRESS.?
Sim
TRATA BOTÃO 1
Não
BOTÃO 2 PRESS.?
Sim
TRATA BOTÃO 2
Não
BOTÃO 3 PRESS.?
Sim
TRATA BOTÃO 3
Não

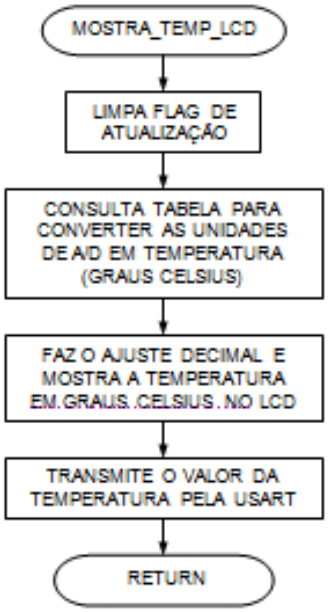

TRATA BOTÃO 0
PODE INC. INTENSIDADE?
Não
Sim
INCREMENTA INTENSIDADE_AQUE
MULTIPLICA INTENSIDADE_AQUE POR 10d E ATUALIZA PWM (CCP1)
FAZ AJUSTE DECIMAL DA INTENSIDADE E MOSTRA VALOR NO LCD
1

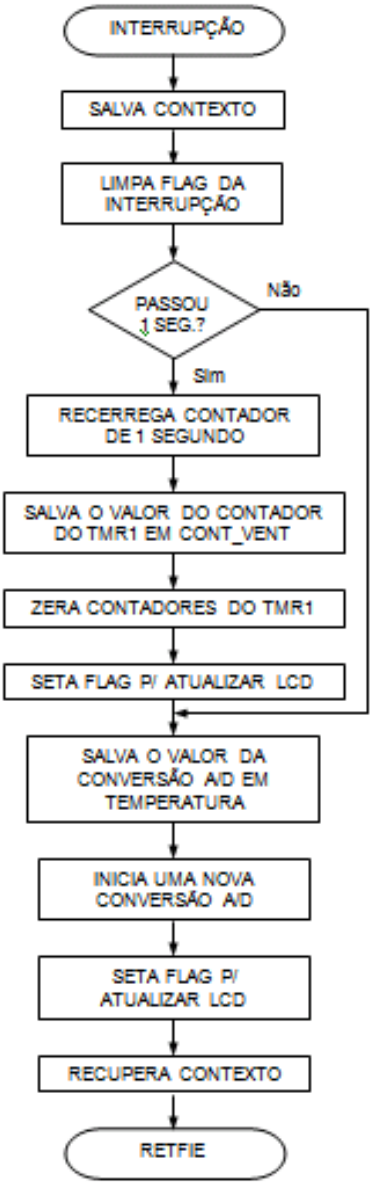

TRATA BOTÃO 1
PODE DEC. INTENSIDADE?
Não
Sim
DECREMENTA INTENSIDADE_AQUE
MULTIPLICA INTENSIDADE_AQUE POR 10d E ATUALIZA PWM (CCP1)
FAZ AJUSTE DECIMAL DA INTENSIDADE E MOSTRA VALOR NO LCD
1


TRATA BOTÃO 2
PODE INC. INTENSIDADE?
Não
Sim
INCREMENTA INTENSIDADE_VENT
MULTIPLICA INTENSIDADE_VENT POR 10d E ATUALIZA PWM (CCP2)
1


TRATA BOTÃO 3
PODE DEC. INTENSIDADE?
Não
Sim
DECREMENTA INTENSIDADE_VENT
MULTIPLICA INTENSIDADE_VENT POR 10d E ATUALIZA PWM (CCP2)
1


MOSTRA_RPS_LCD
LIMPA FLAG DE ATUALIZAÇÃO
CALCULA AS ROTAÇÕES POR SEGUNDO DO VENTILADOR (DIVIDE O CONTADOR PELO NÚMERO DE PALHETAS)
FAZ O AJUSTE DECIMAL E MOSTRA AS ROTAÇÕES NO LCD
RETURN


MOSTRA_TEMP_LCD
LIMPA FLAG DE ATUALIZAÇÃO
CONSULTA TABELA PARA CONVERTER AS UNIDADES DE A/D EM TEMPERATURA (GRAUS CELSIUS)
FAZ O AJUSTE DECIMAL E MOSTRA A TEMPERATURA EM GRAUS CELSIUS NO LCD
TRANSMITE O VALOR DA TEMPERATURA PELA USART
RETURN


INTERRUPÇÃO
SALVA CONTEXTO
LIMPA FLAG DA INTERRUPÇÃO
PASSOU 1 SEG.?
Não
Sim
RECERREGA CONTADOR DE 1 SEGUNDO
SALVA O VALOR DO CONTADOR DO TMR1 EM CONT_VENT
ZERA CONTADORES DO TMR1
SETA FLAG P/ ATUALIZAR LCD
SALVA O VALOR DA CONVERSÃO A/D EM TEMPERATURA
INICIA UMA NOVA CONVERSÃO A/D
SETA FLAG P/ ATUALIZAR LCD
RECUPERA CONTEXTO
RETFIE

22.5. Código

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Exemplo de Código para utilização com a McMaster             *
 *                                 Exemplo 20                                *
 *                                                                           *
 *                                  MOSAICO                                  *
 *                                                                           *
 *  TEL: (0XX11) 4992-8775                 SITE: www.mosaico.com.br          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                            *
 *   DATA : 18/01/2010                                                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              Descrição geral                              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*
ESTE EXEMPLO FOI ELABORADO PARA EXPLICAR O FUNCIONAMENTO DO TMR1 COMO
CONTADOR, UTILIZADO NA PLACA EXP01 PARA CONTAR AS ROTAÇÕES DO VENTILADOR.
O SOFTWARE CONVERTE O CANAL 0 DO CONVERSOR A/D (SENSOR DE TEMPERATURA).
DOIS PWMs FORAM UTILIZADOS, UM PARA MODULAR A RESISTÊNCIA DE AQUECIMENTO
E OUTRO PARA A VELOCIDADE DO VENTILADOR.
COM AS TECLAS S1 E S2 PODE-SE VARIAR O PWM DO AQUECEDOR E COM AS TECLAS
S3 E S4 O PWM DO VENTILADOR.
NO LCD SÃO MOSTRADOS OS VALORES DO PWM DO AQUECEDOR, O NÚMERO DE ROTAÇÕES
POR SEGUNDO DO VENTILADOR E A TEMPERATURA DO DIODO JÁ CONVERTIDA EM GRAUS
CELSIUS. ALÉM DISSO, O VALOR ATUAL DA TEMPERATURA DO DIODO É TRANSMITIDO
PERIODICAMENTE ATRAVÉS DA USART.
*/
/*      CONFIGURAÇÂO DOS JUMPERS DA PLACA:

        CAP USB(RC3)                -> ON
        DATA I2C                    -> OFF
        CLOCK I2C                   -> OFF
        COLUNA 2(RB1)               -> ON
        COLUNA 1(RB0)               -> ON
        LINHA 1 / DISPLAY MILHAR(RB4)    -> ON
        LEDS ESPECIAS (RC0,RC1,RC2)      -> ON
*/


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                         DEFINIÇÃO PIC                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <p18F4550.h>             // Register definitions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                INCLUDES DAS FUNÇÕES DE PERIFÉRICOS DO PIC                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <pwm.h>                   //PWM library functions
#include <adc.h>                    //ADC library functions
#include <timers.h>                 //Timer library functions
#include <delays.h>                 //Delay library functions
#include <i2c.h>                    //I2C library functions
#include <stdlib.h>                 //Library functions
#include <usart.h>                  //USART library functions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
 *                          Configurações para gravação                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#pragma config FOSC = HS
#pragma config CPUDIV = OSC1_PLL2
#pragma config WDT = OFF
#pragma config WDTPS = 128
#pragma config LVP = OFF
#pragma config PWRT = ON
#pragma config BOR = OFF
#pragma config BORV = 0
#pragma config PBADEN = OFF
#pragma config DEBUG = OFF
#pragma config PLLDIV = 1
#pragma config USBDIV = 1
#pragma config FCMEN = OFF
#pragma config IESO = OFF
#pragma config VREGEN = OFF
#pragma config MCLRE = ON
#pragma config LPT1OSC = OFF
#pragma config CCP2MX = ON
#pragma config STVREN = OFF
#pragma config ICPRT = OFF

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Definição e inicialização das variáveis Globais          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.

unsigned char    FILTRO;
unsigned char    tempo_turbo = 30;
unsigned char    unidade;
unsigned char    dezena;
unsigned char    centena;
unsigned char    tempo = 100;
unsigned char    temperatura = 0;
unsigned short   cont_vent = 0;
unsigned char    intensidade_vent = 0;
unsigned char    intensidade_aquec = 0;

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Constantes internas                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de constantes facilita a programação e a manutenção.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *            TABELA DE LINEARIZAÇÃO DO SENSOR DE TEMPERATURA            *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
rom const unsigned char tabela_temp[256] = {
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,                                  //15
0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,                                  //31
2,2,3,3,4,4,5,5,6,6,7,7,8,8,9,9,                                  //47
10,10,11,11,12,12,13,13,14,14,15,15,16,16,17,17,                  //63
18,18,19,19,20,20,21,21,22,22,23,23,23,24,24,25,                  //79
25,26,26,27,27,28,28,29,29,30,30,31,31,32,32,33,                  //95
33,34,34,35,35,36,36,37,37,38,38,39,39,40,40,41,                  //111
41,42,42,43,43,44,44,45,45,46,46,47,47,48,48,49,                  //127
49,50,50,51,51,52,52,53,53,54,54,55,55,56,56,57,                  //143
57,58,58,59,59,60,60,61,61,62,62,63,63,64,64,65,                  //159
65,66,66,67,67,68,68,69,69,70,70,71,71,72,72,73,                  //175
73,74,74,75,75,76,76,77,77,78,78,79,79,80,80,81,                  //191
81,82,82,83,83,84,84,85,85,86,86,87,87,88,88,089,                 //207
89,90,90,91,91,92,92,93,93,94,94,95,95,96,96,097,                 //223
97,98,98,99,99,100,100,101,101,102,102,103,103,104,104,104,       //239
105,105,106,106,107,107,108,108,109,109,110,110,111,111,112,112   //255
};
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Declaração dos flags de software                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de flags ajuda na programação e economiza memória RAM.

struct
{
    unsigned BIT0:1;
    unsigned BIT1:1;
    unsigned BIT2:1;
    unsigned BIT3:1;
    unsigned BIT4:1;
    unsigned BIT5:1;
    unsigned BIT6:1;
    unsigned BIT7:1;
}FLAGSbits;            //ARMAZENA OS FLAGS DE CONTROLE

#define    MOSTRA_RPS            FLAGSbits.BIT0    //ATUALIZAÇÃO DA LEITURA DA ROTAÇÃO
#define    MOSTRA_TEMPERATURA    FLAGSbits.BIT1    //ATUALIZAÇÃO DA LEITURA DA
TEMPERATURA

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                 Definição e inicialização dos port's                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                     PROTOTIPAGEM DE FUNÇÕES                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void comando_lcd(unsigned char caracter);
void escreve_lcd(unsigned char caracter);
void limpa_lcd(void);
void inicializa_lcd(void);
void escreve_frase(const rom char *frase);
void tela_principal(void);
void TRATA_INT_TIMER2(void);
void TRATA_HIGH_INT(void);

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                                ENTRADAS                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// As entradas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define BT1 PORTBbits.RB0       //BOTÃO 1
                                //0 -> PRESSIONADO
                                //1 -> LIBERADO

#define BT2 PORTBbits.RB1       //BOTÃO 2
                                //0 -> PRESSIONADO
                                //1 -> LIBERADO

#define BT3 PORTBbits.RB2       //BOTÃO 3
                                //0 -> PRESSIONADO
                                //1 -> LIBERADO

#define BT4 PORTBbits.RB3       //BOTÃO 4
                                //0 -> PRESSIONADO
                                //1 -> LIBERADO

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                                SAÍDAS                                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//As saídas devem ser associadas a nomes para facilitar a programação e
```

```
//futuras alterações do hardware.

#define     LINHA_4     PORTBbits.RB7   //PINO PARA ATIVAR LINHA 4 DO TECLADO
                                        //MATRICIAL
                                        //1 -> LINHA ATIVADA
                                        //0 -> LINHA DESATIVADA

#define rs        PORTEbits.RE0    // via do lcd que sinaliza recepção de dados ou
comando
#define enable    PORTEbits.RE1    // enable do lcd

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               Rotina que envia um COMANDO para o LCD                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void comando_lcd(unsigned char caracter)
{
    rs = 0;                         // seleciona o envio de um comando
    PORTD = caracter;               // carrega o PORTD com o caracter
    enable = 1 ;                    // gera pulso no enable
    Delay10TCYx(1);                 // espera 10 microsegundos
    enable = 0;                     // desce o pino de enable
    Delay10TCYx(4);                 // espera mínimo 40 microsegundos
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *            Rotina que envia um DADO a ser escrito no LCD                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void escreve_lcd(unsigned char caracter)
{
    rs = 1;                     // seleciona o envio de um comando
    PORTD = caracter;           // carrega o PORTD com o caracter
    enable = 1;                 // gera pulso no enable
    Delay10TCYx(1);             // espera 10 microsegundos
    enable = 0;                 // desce o pino de enable
    Delay10TCYx(4);             // espera mínimo 40 microsegundos
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função para limpar o LCD                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void limpa_lcd(void)
{
    comando_lcd(0x01);            // limpa lcd
    Delay1KTCYx(2);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Inicialização do Display de LCD                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void inicializa_lcd(void)
{
    comando_lcd(0x30);            // envia comando para inicializar display
    Delay1KTCYx(4);               // espera 4 milisengundos

    comando_lcd(0x30);            // envia comando para inicializar display
    Delay10TCYx(10);              // espera 100 microsengundos

    comando_lcd(0x30);            // envia comando para inicializar display

    comando_lcd(0x38);            // liga o display, sem cursor e sem blink

    limpa_lcd();                  // limpa lcd
```

```
    comando_lcd(0x0c);            // display sem cursor

    comando_lcd(0x06);            // desloca cursor para a direita
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *            Função para escrever uma frase no LCD                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void escreve_frase(const rom char *frase)
{
    do
    {
        escreve_lcd(*frase);
    }while(*++frase);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      FUNÇÃO PARA SEPARAR DÍGITOS                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void converte_bcd(unsigned char aux)
{
    unidade = 0;
    dezena = 0;
    centena = 0;
    if (aux == 0)return;    //SE AUX. FOR IGUAL A ZERO, SAI
    while(aux--)
    {
        unidade++;
        if(unidade != 10)continue;
        unidade = 0;
        dezena++;
        if (dezena != 10)continue;
        dezena = 0;
        centena++;
        if (centena != 10)continue;
        centena = 0;
    }
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Função de tratamento de interrupção de Timer2              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void TRATA_INT_TIMER2(void)
{
    PIR1bits.TMR2IF = 0;        // limpa flag de interrupção
    tempo--;                    // DECREMENTA TEMPO
    if(!tempo)                  // TEMPO = 0?
    {
        tempo = 100;            // REINICIA TEMPO
        cont_vent = ReadTimer1();    //LER O VALOR DO TIMER1
        WriteTimer1(0);              //carraga timer1 com 0
        MOSTRA_RPS = 1;              // SETA FLAG MOSTRA_RPS
    }
    temperatura = ADRESH;        // LER O RESULTADO DA CONVERSÃO AD
    ConvertADC();                // INICIA UMA NOVA CONVERSÃO
    MOSTRA_TEMPERATURA = 1;      // SETA FLAG MOSTRA_TEMPERATURA
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função Principal                              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
```

```
void main ()
{
    PORTA = 0x00;               //Limpa PORTA
    PORTB = 0x00;               //Limpa PORTB
    PORTC = 0x00;               //Limpa PORTC
    PORTD = 0x00;               //Limpa PORTD
    PORTE = 0x00;               //Limpa PORTE

    LATA = 0x00;                //Limpa PORTA
    LATB = 0x00;                //Limpa PORTB
    LATC = 0x00;                //Limpa PORTC
    LATD = 0x00;                //Limpa PORTD
    LATE = 0x00;                //Limpa PORTE

    TRISA = 0b11101111;                 //CONFIG DIREÇÃO DOS PINOS PORTA0
    TRISB = 0b00001111;                 //CONFIG DIREÇÃO DOS PINOS PORTB
    TRISC = 0b10111001;                 //CONFIG DIREÇÃO DOS PINOS PORTC
    TRISD = 0b00000000;                 //CONFIG DIREÇÃO DOS PINOS PORTD
    TRISE = 0b00000100;                 //CONFIG DIREÇÃO DOS PINOS PORTE

    OpenADC(ADC_FOSC_8 & ADC_LEFT_JUST & ADC_0_TAD, ADC_CH1 & ADC_INT_OFF &
ADC_VREFPLUS_VDD
            & ADC_VREFMINUS_VSS,12);
    //CONFIGURAÇÃO DO AD
    OpenUSART(USART_TX_INT_OFF  &  USART_RX_INT_OFF  &  USART_ASYNCH_MODE &
              USART_EIGHT_BIT   &  USART_CONT_RX     &  USART_BRGH_HIGH,25);
    //CONFIGURAÇÃO DA USART

    OpenTimer1(TIMER_INT_OFF & T1_SOURCE_EXT & T1_16BIT_RW & T1_PS_1_1 & T1_OSC1EN_OFF
&
               T1_SYNC_EXT_ON);
    //CONFIGURAÇÃO DO TIMER1

    OpenTimer2(TIMER_INT_ON & T2_PS_1_4 & T2_POST_1_10);
    //CINFIGURAÇÃO DO TIMER2

//    OpenPWM1(249);
    //CONFIGURAÇÃO DO MÓDULO CCP1 -> PWM
    CCP1CON = 0B00001100;
    PR2 = 249;

//    OpenPWM2(249);
    //CONFIGURAÇÃO DO MÓDULO CCP2 -> PWM
    CCP2CON = 0B00001111;

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Inicialização do Sistema                         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

    LINHA_4 = 1;                //ATIVA LINHA 4 DO TECLADO MATRICIAL

    SetDCPWM1(0);    //DUTY CYCLE DO PWM1 EM 0
    SetDCPWM2(0);    //DUTY CYCLE DO PWM2 EM 0

    INTCONbits.PEIE = 1;    //LIGA AS INTERRUPÇÕES
    INTCONbits.GIE = 1;

    inicializa_lcd();    //INICIALIZA O LCD

    MOSTRA_RPS = 0;         //LIMPA OS FLAGS
    MOSTRA_TEMPERATURA = 0;

    comando_lcd(0x80);    //IMPRIME A TELA PRINCIPAL
    escreve_frase("Aquec. RPS TEMP");
    comando_lcd(0xC0);
    escreve_frase(" 000%  000 000 C");
```

```
    comando_lcd(0xCE);
    escreve_lcd(0xDF);

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Rotina Principal                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

    while(1)
    {
        ClrWdt();

        if(MOSTRA_RPS)                      //DEVE MOSTRAR O VALOR DO RPS?
        {
            MOSTRA_RPS = 0;                 //LIMPA FLAG
            cont_vent = cont_vent / 7;      //DIVIDE A LEITURA DO TIMER1 POR 7
            converte_bcd((unsigned char)cont_vent);//EFETUA A CONVERSÃO EM BCD
            comando_lcd(0xC7);              // posiciona o cursor na linha 1, coluna 7
            escreve_lcd (centena + 0x30);  // imprime mensagem no lcd
            escreve_lcd (dezena + 0x30);
            escreve_lcd (unidade + 0x30);
        }

        if(MOSTRA_TEMPERATURA)              //DEVE MOSTRAR A TEMPERATURA?
        {
            MOSTRA_TEMPERATURA = 0;         //LIMPA FLAG
            converte_bcd(tabela_temp[temperatura]);//BUSCA O VALOR CORRETO DA TEMP.
                                                   //E EFETUA A CONVERSÃO EM BCD
            comando_lcd(0xCB);              // posiciona o cursor na linha 1, coluna 11
            escreve_lcd (centena + 0x30); // imprime mensagem no lcd
            escreve_lcd (dezena + 0x30);
            escreve_lcd (unidade + 0x30);

            while(BusyUSART());            //AGUARDA BUFFER DE TRANSMISSÃO FICAR VAZIO
            WriteUSART(temperatura);       //ENVIA TEMPERATURA À USART
        }

        if(BT1)                             //BOTÃO 1 PRESSIONADO?
        {
            FILTRO--;                       //DECREMENTA FILTRO DE BOTÃO
            if (!FILTRO)                    //FILTRO = 0?
            {
                tempo_turbo--;              //DECREMENTA TURBO
                if (!tempo_turbo)           //TURBO =0?
                {
                    if(intensidade_aquec < 100)
                            {
                                    intensidade_aquec++;
                     //SE INTENSIDADE_AQUEC < 100, INCREMENTA
                     converte_bcd(intensidade_aquec);  //CONVERTE INTENSIDADE_AQUE EM
BCD
                     comando_lcd(0xC1);                // posiciona o cursor na linha
1, coluna 1
                     escreve_lcd (centena + 0x30);     // imprime mensagem no lcd
                     escreve_lcd (dezena + 0x30);
                     escreve_lcd (unidade + 0x30);
                     SetDCPWM1((unsigned short)intensidade_aquec*10);//ATUALIZA PWM1
                     tempo_turbo = 30;    //REINICIA TURBO DE TECLA
                            }
                }
            }
            continue;
        }

        if(BT2)                           //BOTÃO 2 PRESSIONADO?
        {
            FILTRO--;                     //DECREMENTA FILTRO DE BOTÃO
```

```
            if (!FILTRO)                //FILTRO = 0?
            {
                tempo_turbo--;          //DECREMENTA TURBO
                if (!tempo_turbo)       //TURBO =0?
                {
                    if(intensidade_aquec > 0)
                            {
                            intensidade_aquec--;
                    //SE INTENSIDADE_AQUEC > 0, DECREMENTA
                    converte_bcd(intensidade_aquec);    //CONVERTE INTENSIDADE_AQUE EM
BCD
                    comando_lcd(0xC1);                  // posiciona o cursor na linha
1, coluna 1
                    escreve_lcd (centena + 0x30);       // imprime mensagem no lcd
                    escreve_lcd (dezena + 0x30);
                    escreve_lcd (unidade + 0x30);
                    SetDCPWM1((unsigned short)intensidade_aquec*10);//ATUALIZA PWM1
                    tempo_turbo = 30;    //REINICIA TURBO DE TECLA
                            }
                }
            }
            continue;
        }

        if(BT3)                         //BOTÃO 3 PRESSIONADO?
        {
            FILTRO--;                   //DECREMENTA FILTRO DE BOTÃO
            if (!FILTRO)                //FILTRO = 0?
            {
                tempo_turbo--;          //DECREMENTA TURBO
                if (!tempo_turbo)       //TURBO =0?
                {
                    if(intensidade_vent < 100) intensidade_vent++;
                    //SE INTENSIDADE_VENT < 100, INCREMENTA
                    SetDCPWM2((unsigned short)intensidade_vent*10);//ATUALIZA PWM2
                    tempo_turbo = 30;    //REINICIA TURBO DE TECLA
                }
            }
            continue;
        }

        if(BT4)                         //BOTÃO 4 PRESSIONADO?
        {
            FILTRO--;                   //DECREMENTA FILTRO DE BOTÃO
            if (!FILTRO)                //FILTRO = 0?
            {
                tempo_turbo--;          //DECREMENTA TURBO
                if (!tempo_turbo)       //TURBO =0?
                {
                    if(intensidade_vent > 0) intensidade_vent--;
                    //SE INTENSIDADE_VENT > 0, DECREMENTA
                    SetDCPWM2((unsigned short)intensidade_vent*10);//ATUALIZA PWM2
                    tempo_turbo = 30;    //REINICIA TURBO DE TECLA
                }
            }
            continue;
        }
        FILTRO = 255;//REINICIA FILTRO DE BOTÃO
    }
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
// *             ROTINA DE TRATAMENTO DE INT DE ALTA PRIORIDADE              *
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

#pragma code VETOR_HIGH_PRIORITY = 0x0008
```

```
void HIGH_int(void)
{
  _asm goto TRATA_HIGH_INT _endasm
}
#pragma code

#pragma interrupt TRATA_HIGH_INT
void TRATA_HIGH_INT(void)
{
    if(PIR1bits.TMR2IF) TRATA_INT_TIMER2();
}
```

22.6. Dicas e Comentários

Para converter a temperatura do diodo lida pelo conversor A/D na temperatura ambiente em graus Celsius utilizou-se uma tabela de conversão, do mesmo tipo da utilizada para converter um número BCD numa representação gráfica para displays de 7 segmentos. Porém, como a tabela é relativamente extensa (possui 256 posições, relativos aos 8 bits da conversão A/D), para economizar espaço na listagem do software utilizou-se a diretriz DT. O compilador interpreta a diretriz substituindo cada valor que a sucede por uma instrução RETLW. Assim, o código fonte é exatamente o mesmo que se fosse utilizada a instrução RETLW, porém, ao invés de escrever 256 vezes a instrução RETLW e o valor de retorno, utilizou-se a diretriz. Em cada linha foram passados 16 argumentos, desta forma, no exemplo da experiência 20, o compilador substitui cada linha com a instrução DT por 16 instruções RETLW. Este recurso não altera em nada o código fonte e o funcionamento do programa.

22.7. Exercícios Propostos

1. Fazer um software para controlar a temperatura do diodo. Este software deve medir a temperatura e alterar o valor do PWM da resistência para manter a temperatura do diodo em constante em 50ºC.
2. Fazer um software para controlar a velocidade de rotação do ventilador. Este software deve medir rotação e alterar o valor do PWM a fim de manter a rotação constante.

23. Exemplo placa de experiência EXP01

23.1. Objetivo

O objetivo desta experiência é validar a placa de experiência EXP01.

23.2. Descrição

Esta placa de experiência possui um sistema taco-gerador, um ventilador e um sistema de aquecimento resistivo. Para trabalhar com esses recursos carregue o exemplo 20 desenvolvido para o respectivo microcontrolador.

Obs: (1) O acionamento da lâmpada é possível apenas quando trabalhamos com os microcontroladores PIC16F877A e PIC18F4520; o microcontrolador PIC18F4550 reserva esta saída para o módulo USB interno.

24. Exemplo placa de experiência EXP02

24.1. Objetivo

O objetivo desta experiência é validar a placa de experiência EXP02.

24.2. Descrição

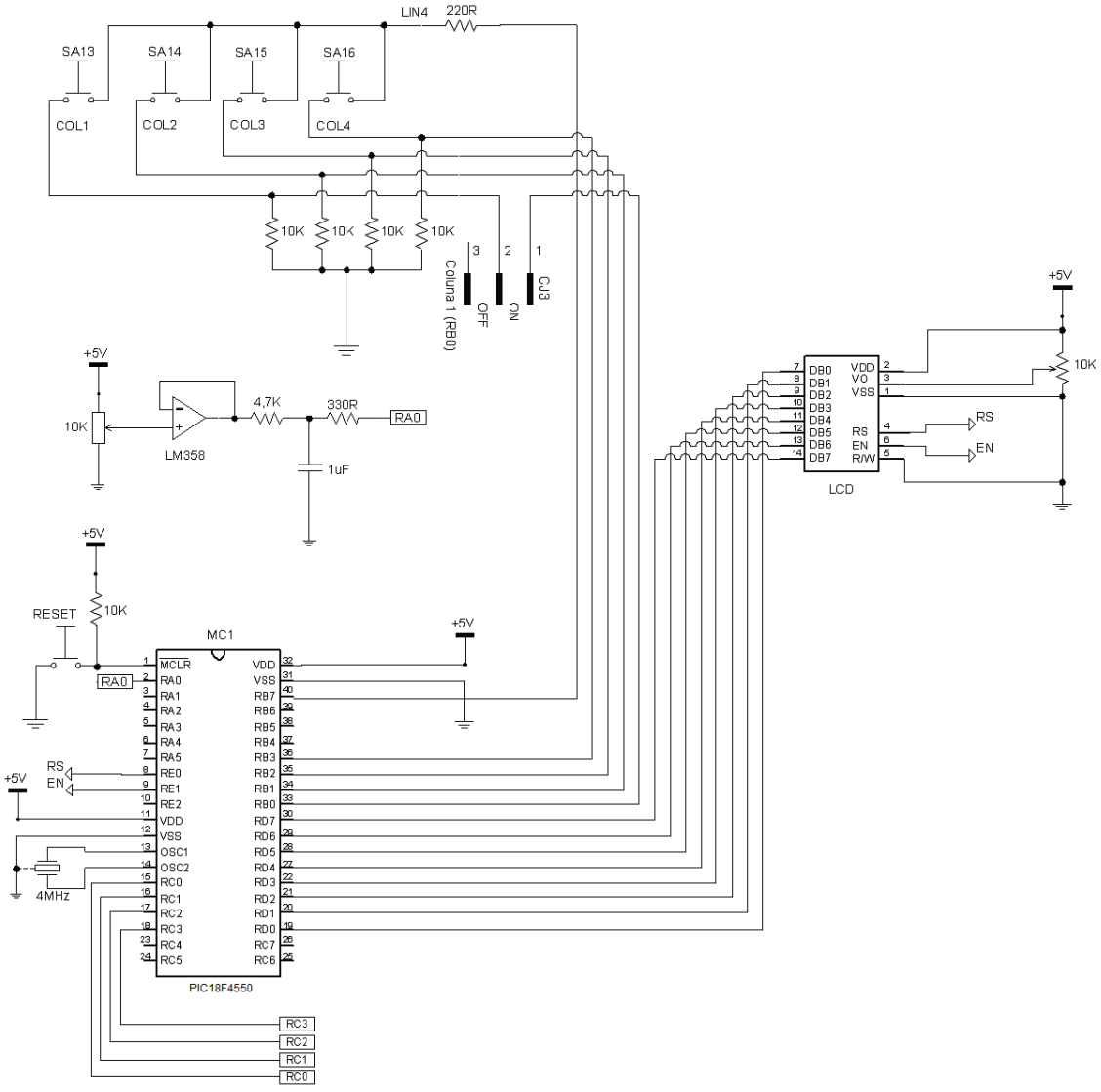
Este PIC possui 2 canais de PWMs (CCP1 e CCP2), cada um com resolução máxima de 10 bits. Cada saída está ligada a um integrador cuja saída é diretamente proporcional ao ciclo ativo do PWM. A variação do ciclo ativo do PWM será feita pelo potenciômetro de 10k que se encontra na placa principal via conversor A/D. O valor da saída do conversor D/A é mostrado em percentual (de 0% a 100%).

Alem disso é testado os relés. Pressionando o botão 0 o relé 1 atraca até liberar o mesmo botão. Pressionando o botão 1 o relé 2 atraca até liberar o mesmo botão.

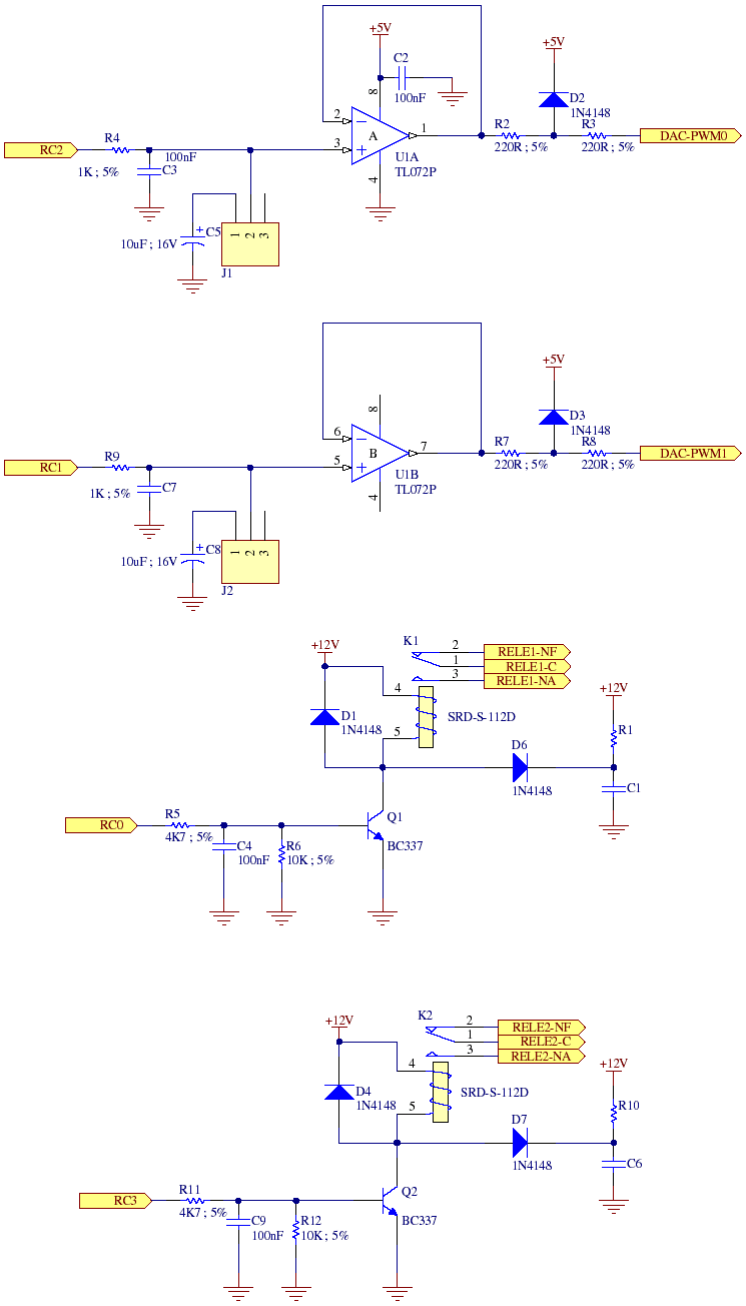
Obs: (1) O acionamento do relé 2 é possível apenas quando trabalhamos com os microcontroladores PIC16F877A e PIC18F4520; o microcontrolador PIC18F4550 reserva esta saída para o módulo USB interno.

24.3. Esquema Elétrico

24.3.1. Placa principal



24.3.2. Placa de experiências



24.4. Código

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Exemplo de Código para utilização com a McMaster            *
 *                                EXP02                                    *
 *                                                                         *
 *                               MOSAICO                                   *
 *                                                                         *
 *  TEL: (0XX11) 4992-8775                SITE: www.mosaico.com.br         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                          *
 *   DATA : 18/01/2010                                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            Descrição geral                              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*
   ESTE EXEMPLO FOI ELABORADO PARA TRABALHAR COM A PLACA DE EXPERIMENTOS 2
*/
/*      CONFIGURAÇÂO DOS JUMPERS DA PLACA:

        CAP USB(RC3)              -> ON
        DATA I2C                  -> OFF
        CLOCK I2C                 -> OFF
        COLUNA 2(RB1)             -> ON
        COLUNA 1(RB0)             -> ON
        LINHA 1 / DISPLAY MILHAR(RB4)    -> ON
        LEDS ESPECIAS (RC0,RC1,RC2)      -> ON

*/
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      DEFINIÇÃO PIC                      *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <p18F4550.h>            // Register definitions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               INCLUDES DAS FUNÇÕES DE PERIFÉRICOS DO PIC              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <pwm.h>                  //PWM library functions
#include <adc.h>                   //ADC library functions
#include <timers.h>                //Timer library functions
#include <delays.h>                //Delay library functions
#include <i2c.h>                   //I2C library functions
#include <stdlib.h>                //Library functions
#include <usart.h>                 //USART library functions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Configurações para gravação                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#pragma config FOSC = HS
#pragma config CPUDIV = OSC1_PLL2
#pragma config WDT = OFF
#pragma config WDTPS = 128
#pragma config LVP = OFF
#pragma config PWRT = ON
#pragma config BOR = OFF
#pragma config BORV = 0
```

```
#pragma config PBADEN = OFF
#pragma config DEBUG = OFF
#pragma config PLLDIV = 1
#pragma config USBDIV = 1
#pragma config FCMEN = OFF
#pragma config IESO = OFF
#pragma config VREGEN = OFF
#pragma config MCLRE = ON
#pragma config LPT1OSC = OFF
#pragma config CCP2MX = ON
#pragma config STVREN = OFF
#pragma config ICPRT = OFF

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Definição e inicialização das variáveis Globais          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.

unsigned int buffer = 0;

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Constantes internas                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de constantes facilita a programação e a manutenção.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *           TABELA DE LINEARIZAÇÃO DO SENSOR DE TEMPERATURA             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Declaração dos flags de software                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de flags ajuda na programação e economiza memória RAM.


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Definição e inicialização dos port's                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       PROTOTIPAGEM DE FUNÇÕES                         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                                ENTRADAS                                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// As entradas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define BT1 PORTBbits.RB0       //BOTÃO 1
                                //0 -> PRESSIONADO
                                //1 -> LIBERADO

#define BT2 PORTBbits.RB1       //BOTÃO 2
                                //0 -> PRESSIONADO
                                //1 -> LIBERADO

#define BT3 PORTBbits.RB2       //BOTÃO 3
                                //0 -> PRESSIONADO
                                //1 -> LIBERADO

#define BT4 PORTBbits.RB3       //BOTÃO 4
                                //0 -> PRESSIONADO
                                //1 -> LIBERADO
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              SAÍDAS                                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//As saídas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define     LINHA_4     LATBbits.LATB7  //PINO PARA ATIVAR LINHA 4 DO TECLADO
                                        //MATRICIAL
                                        //1 -> LINHA ATIVADA
                                        //0 -> LINHA DESATIVADA

#define     RELE1       LATCbits.LATC0  //PINO PARA ATIVAR LINHA 4 DO TECLADO
                                        //MATRICIAL
                                        //1 -> LINHA ATIVADA
                                        //0 -> LINHA DESATIVADA

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               Rotina que envia um COMANDO para o LCD                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */



/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função Principal                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void main ()
{
    PORTA = 0x00;               //Limpa PORTA
    PORTB = 0x00;               //Limpa PORTB
    PORTC = 0x00;               //Limpa PORTC
    PORTD = 0x00;               //Limpa PORTD
    PORTE = 0x00;               //Limpa PORTE

    LATA = 0x00;                //Limpa PORTA
    LATB = 0x00;                //Limpa PORTB
    LATC = 0x00;                //Limpa PORTC
    LATD = 0x00;                //Limpa PORTD
    LATE = 0x00;                //Limpa PORTE

    TRISA = 0b11101111;                 //CONFIG DIREÇÃO DOS PINOS PORTA0
    TRISB = 0b00001111;                 //CONFIG DIREÇÃO DOS PINOS PORTB
    TRISC = 0b10110000;                 //CONFIG DIREÇÃO DOS PINOS PORTC
    TRISD = 0b11111111;                 //CONFIG DIREÇÃO DOS PINOS PORTD
    TRISE = 0b00000100;                 //CONFIG DIREÇÃO DOS PINOS PORTE

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Inicialização do Sistema                      *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

    OpenADC(ADC_FOSC_8 & ADC_RIGHT_JUST & ADC_0_TAD, ADC_CH0 & ADC_INT_OFF &
ADC_REF_VDD_VSS,12);
    //CONFIGURAÇÃO DO AD

        OpenTimer2(TIMER_INT_OFF & T2_PS_1_4 & T2_POST_1_10);
    //CONFIGURAÇÃO DO TIMER2

  PR2 = 249;
    //CONFIGURAÇÃO DO MÓDULO CCP1 -> PWM
    CCP1CON = 0B00001100;

    //CONFIGURAÇÃO DO MÓDULO CCP2 -> PWM
    CCP2CON = 0B00001111;

  LINHA_4 = 1;
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Rotina Principal                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

    while(1)
    {
        if (BT1)                    // Verifica pressionamento do botão1
               RELE1 = 1;           // Aciona rele1
        else
               RELE1 = 0;           // Desaciona rele1

        ConvertADC();       // Realiza uma conversão
        if(BusyADC())       // Verifica se o conversor finalizou conversão
     {
        SetDCPWM1(ReadADC());      // Atualiza canais PWM com valor da
        SetDCPWM2(ReadADC());      // conversão
     }
   }
}
```

25. Exemplo placa de experiência EXP03

25.1. Objetivo

O objetivo desta experiência é validar a placa de experiência EXP03.

25.2. Descrição

A placa de experiência EXP03 oferece como periferia ao microcontrolador uma chave DIP-SWITCH de 8 posições, um potenciômetro digital e um conector USB(1). Seus exemplos estão divididos da seguinte forma:

- Exemplo 3a - leitura das chaves e acendimento dos led´s
- Exemplo 3b - leitura das chaves para comando de potenciômetro digital com leitura da tensão de referência gerada através do módulo A/D exibindo resultado no LCD
- Exemplo 3c - conjunto formado por dois experimentos envolvendo a comunicação USB. (1)Obs: Válidos somente para o microcontrolador PIC18F4550

Os descritivos funcionais dos exemplos USB seguem abaixo:

- Exemplo USB Device - CDC - Basic Demo

  Trata-se de uma demonstração do uso das funções da classe CDC RS232. Depois de gravado, conecte na porta USB e durante a instalação aponte para a pasta Drive USB PIC18F4550.Usando o Hiperterminal do Windows, configure para a porta COM instalada. Pressionando o botão S3 aparecerá a mensagem Button Pressed --. Digitando um caractere, retornará ao terminal o caractere seguinte (por exemplo, se você digitou o 'a', aparecerá 'b').

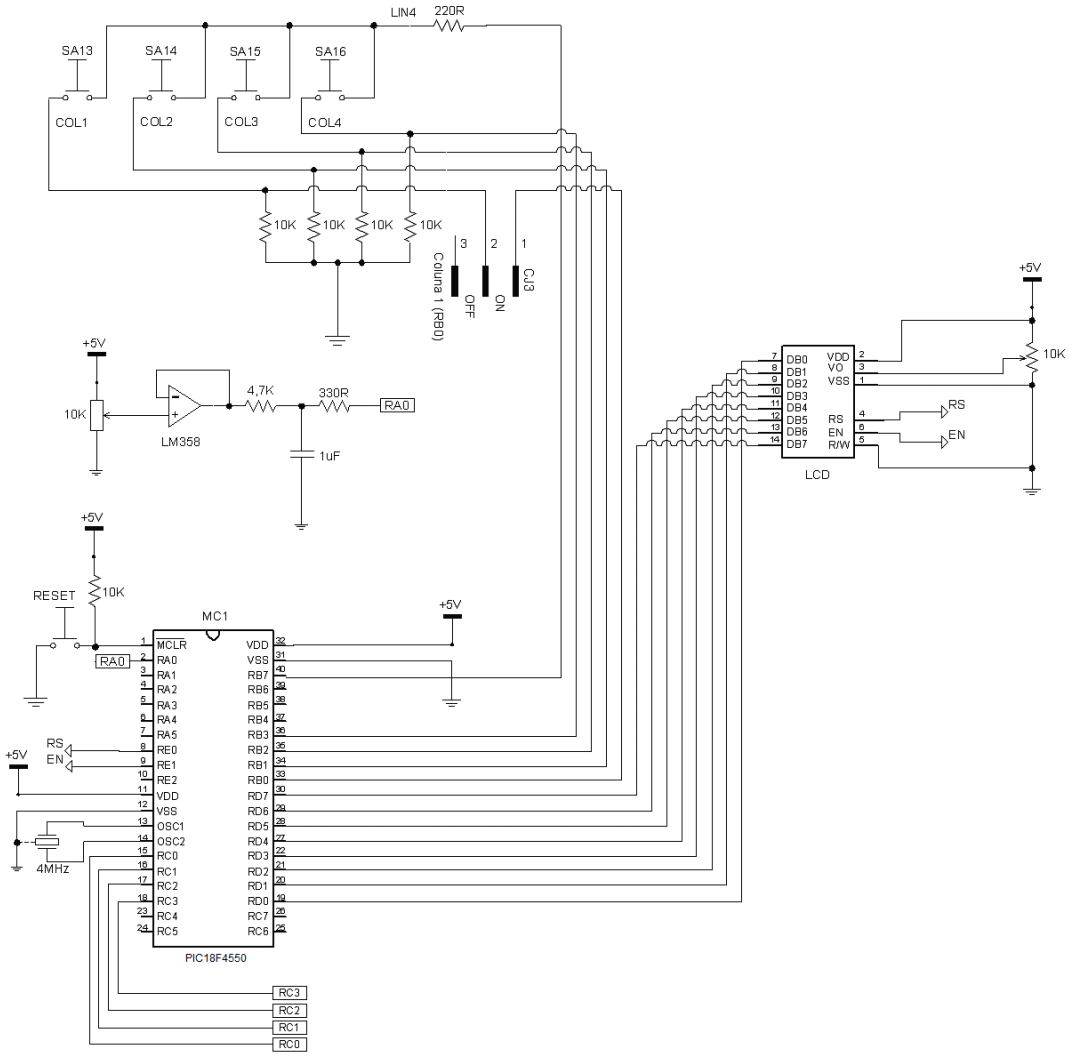
- Exemplo USB Device - HID – Mouse

  Este exemplo é uma demonstração do uso das funções da classe HID. Este exemplo emula um mouse. Ao pressionar o botão S4 o cursor do mouse ficará "em círculo". Pressionando novamente o botão S4, o cursor voltará ao normal.

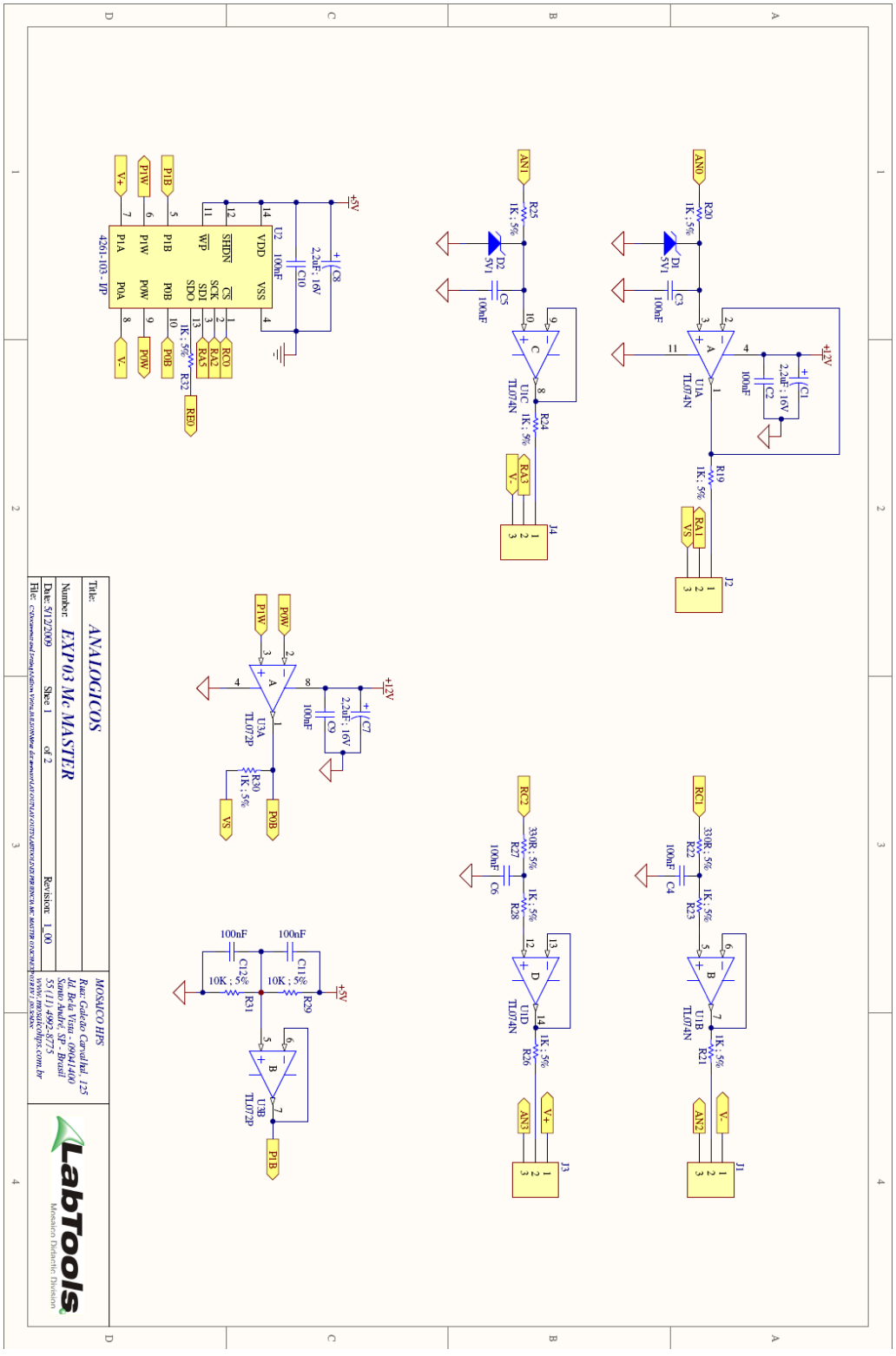
Os dois exemplos estão incluídos na pasta de exemplos do microcontrolador PIC18F4550.

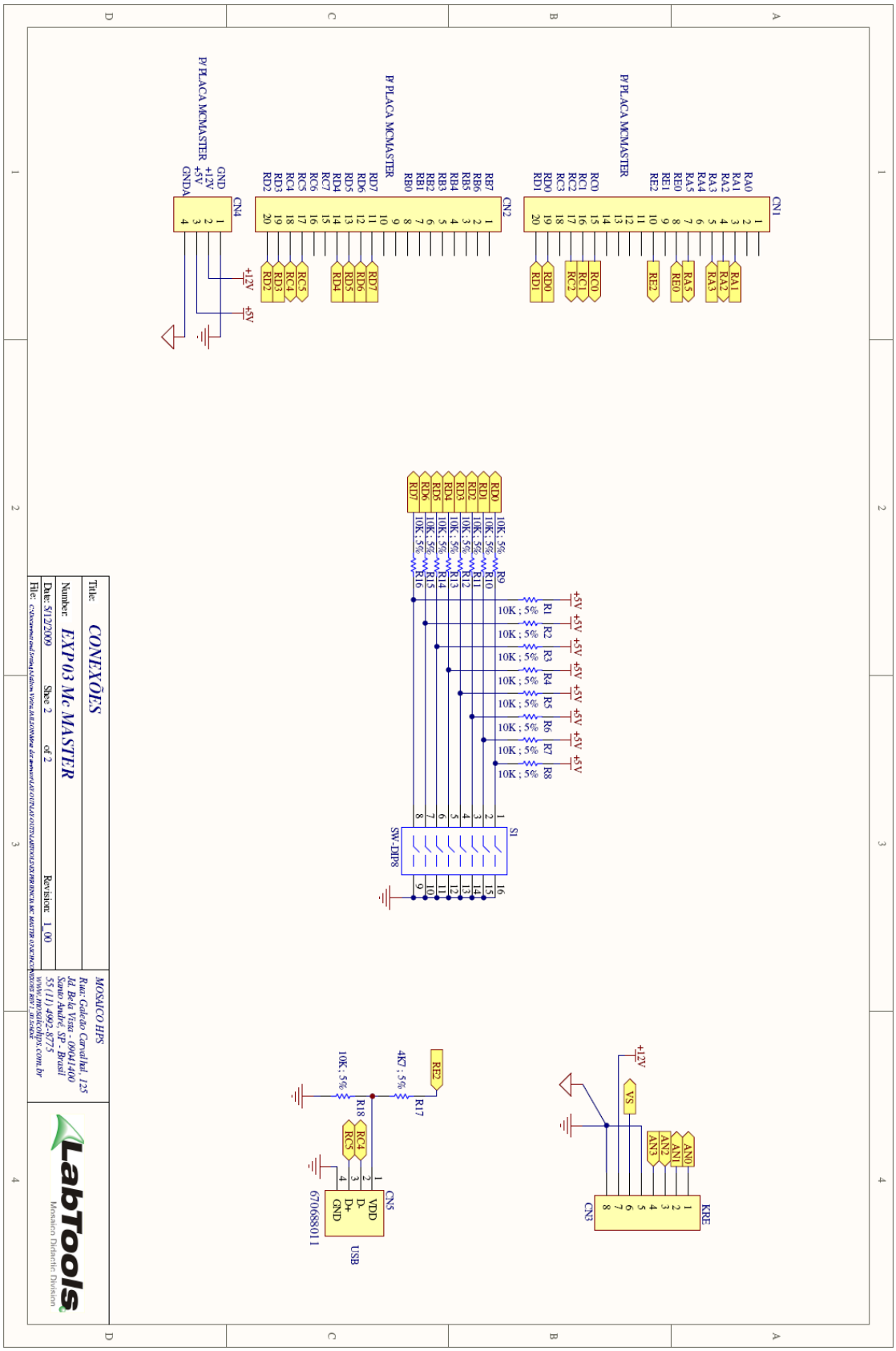
26. Esquema Elétrico

26.1. Placa principal



26.1.1. Placa de experiências




CONEXÕES
EXP03 Mc MASTER
MOSAICO HPS
Rua: Galeão Carvalhal, 125
Jd. Bela Vista - 09041400
Santo André, SP - Brasil
55 (11) 4992-8775
www.mosaicohps.com.br
LabTools
P/ PLACA MCMASTER
CN1
CN2
CN3
CN4
CN5
S1
SW-DIP8
KRE
USB
6706888011
10K ; 5%
4K7 ; 5%

26.2. Código

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Exemplo de Código para utilização com a McMaster            *
 *                              EXP03a                                     *
 *                                                                         *
 *                             MOSAICO                                     *
 *                                                                         *
 *  TEL: (0XX11) 4992-8775                SITE: www.mosaico.com.br         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                          *
 *   DATA : 18/01/2010                                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Descrição geral                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*
ESTE EXEMPLO FOI ELABORADO PARA DEMONSTRAR O FUNCIONAMENTO DO CONJUNTO DE
CHAVES DIP-SWITCH PRESENTES NA PLACA EXP03. CADA CHAVE ACIONADA COMANDA O
ESTADO DE UM LED ASSOCIADO AO PORTD DO MICROCONTROLADOR.
*/
/*      CONFIGURAÇÂO DOS JUMPERS DA PLACA:

        CAP USB(RC3)              -> ON
        DATA I2C                  -> OFF
        CLOCK I2C                 -> OFF
        COLUNA 2(RB1)             -> ON
        COLUNA 1(RB0)             -> ON
        LINHA 1 / DISPLAY MILHAR(RB4)    -> ON
        LEDS ESPECIAS (RC0,RC1,RC2)      -> ON


*/
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       DEFINIÇÃO PIC                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <p18F4550.h>            // Register definitions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               INCLUDES DAS FUNÇÕES DE PERIFÉRICOS DO PIC                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <pwm.h>                  //PWM library functions
#include <adc.h>                   //ADC library functions
#include <timers.h>                //Timer library functions
#include <delays.h>                //Delay library functions
#include <i2c.h>                   //I2C library functions
#include <stdlib.h>                //Library functions
#include <usart.h>                 //USART library functions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Configurações para gravação                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#pragma config FOSC = HS
#pragma config CPUDIV = OSC1_PLL2
#pragma config WDT = OFF
#pragma config WDTPS = 128
#pragma config LVP = OFF
#pragma config PWRT = ON
```

```
#pragma config BOR = OFF
#pragma config BORV = 0
#pragma config PBADEN = OFF
#pragma config DEBUG = OFF
#pragma config PLLDIV = 1
#pragma config USBDIV = 1
#pragma config FCMEN = OFF
#pragma config IESO = OFF
#pragma config VREGEN = OFF
#pragma config MCLRE = ON
#pragma config LPT1OSC = OFF
#pragma config CCP2MX = ON
#pragma config STVREN = OFF
#pragma config ICPRT = OFF

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Definição e inicialização das variáveis Globais           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.

unsigned char leds_temp = 0;
unsigned char chaves_temp = 0;
unsigned char i = 0;

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Constantes internas                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de constantes facilita a programação e a manutenção.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Declaração dos flags de software                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de flags ajuda na programação e economiza memória RAM.


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                 Definição e inicialização dos port's                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      PROTOTIPAGEM DE FUNÇÕES                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                               ENTRADAS                                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// As entradas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                               SAÍDAS                                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//As saídas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define C_LEDS    LATAbits.LATA4  //PINO PARA ATIVAR BARRA DE LEDS
                                  //1 -> BARRA ATIVADA
                                  //0 -> BARRA DESATIVADA
#define CHAVES   PORTD

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função Principal                             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
```

```
void main ()
{
    PORTA = 0x00;               //Limpa PORTA
    PORTB = 0x00;               //Limpa PORTB
    PORTC = 0x00;               //Limpa PORTC
    PORTD = 0x00;               //Limpa PORTD
    PORTE = 0x00;               //Limpa PORTE

    LATA = 0x00;                //Limpa PORTA
    LATB = 0x00;                //Limpa PORTB
    LATC = 0x00;                //Limpa PORTC
    LATD = 0x00;                //Limpa PORTD
    LATE = 0x00;                //Limpa PORTE

    TRISA = 0b11101111;                 //CONFIG DIREÇÃO DOS PINOS PORTA0
    TRISB = 0b00001111;                 //CONFIG DIREÇÃO DOS PINOS PORTB
    TRISC = 0b10111000;                 //CONFIG DIREÇÃO DOS PINOS PORTC
    TRISD = 0b11111111;                 //CONFIG DIREÇÃO DOS PINOS PORTD
    TRISE = 0b00000111;                 //CONFIG DIREÇÃO DOS PINOS PORTE

 ADCON1 = 0b00001111;

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Inicialização do Sistema                      *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Rotina Principal                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

    while(1)
    {
        C_LEDS = 0;         // Desabilita a a barra de leds associada ao
                            // aos pinos do PORTD

        TRISD = 0xFF;       // Configura PORTD do microcontrolador como
                            // entrada

        Delay1KTCYx(1);             // Atraso para acomodação do PORT

        leds_temp = PORTD;  // Realiza leitura de todas as chaves
                            // e armazena na variável auxiliar

        Delay1KTCYx(1);             // Atraso para acomodação do PORT

        TRISD = 0x00;       // Configura PORTD do microcontrolador como
                            // saída

        Delay1KTCYx(1);             // Atraso para acomodação do PORT

        LATD = leds_temp;   // Realiza escrita no PORTD do valor
                            // correspondente as chaves acionadas

        C_LEDS = 1;         // Desabilita a a barra de leds associada ao
                            // aos pinos do PORTD

        Delay1KTCYx(1);             // Atraso para exibição dos valores das chaves
                            // através da barra de leds
 }
}
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Exemplo de Código para utilização com a McMaster            *
 *                              EXP03b                                     *
 *                                                                         *
 *                             MOSAICO                                     *
 *                                                                         *
 *  TEL: (0XX11) 4992-8775                SITE: www.mosaico.com.br         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                          *
 *   DATA : 18/01/2010                                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Descrição geral                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*
ESTE EXEMPLO FOI ELABORADO PARA DEMONSTRAR O FUNCIONAMENTO DO POTENCIOMETRO
DIGITAL PRESENTE NA PLACA EXP03. CADA CHAVE ACIONADA DETERMINA UM VALOR
DE RESISTÊNCIA DO POTENCIOMETRO DIGITAL ALTERANDO A TENSÃO DE REFERÊNCIA
MEDIDA CONTINUAMENTE PELO CONVERSOR A/D EXIBIDA NO LCD.
*/
/*      CONFIGURAÇÃO DOS JUMPERS DA PLACA:

        CAP USB(RC3)              -> ON
        DATA I2C                  -> OFF
        CLOCK I2C                 -> OFF
        COLUNA 2(RB1)             -> ON
        COLUNA 1(RB0)             -> ON
        LINHA 1 / DISPLAY MILHAR(RB4)    -> ON
        LEDS ESPECIAS (RC0,RC1,RC2)      -> OFF

/*      CONFIGURAÇÃO DOS JUMPERS DA EXP03:

        J1     -> POSIÇÃO 1,2
        J2     -> POSIÇÃO 2,3
        J3     -> POSIÇÃO 1,2
        J4     -> POSIÇÃO 1,2
*/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        DEFINIÇÃO PIC                                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <p18F4550.h>             // Register definitions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                INCLUDES DAS FUNÇÕES DE PERIFÉRICOS DO PIC               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <pwm.h>                    //PWM library functions
#include <adc.h>                    //ADC library functions
#include <timers.h>                 //Timer library functions
#include <delays.h>                 //Delay library functions
#include <i2c.h>                    //I2C library functions
#include <stdlib.h>                 //Library functions
#include <usart.h>                  //USART library functions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Configurações para gravação                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#pragma config FOSC = HS
#pragma config CPUDIV = OSC1_PLL2
#pragma config WDT = OFF
#pragma config WDTPS = 128
#pragma config LVP = OFF
```

```
#pragma config PWRT = ON
#pragma config BOR = OFF
#pragma config BORV = 0
#pragma config PBADEN = OFF
#pragma config DEBUG = OFF
#pragma config PLLDIV = 1
#pragma config USBDIV = 1
#pragma config FCMEN = OFF
#pragma config IESO = OFF
#pragma config VREGEN = OFF
#pragma config MCLRE = ON
#pragma config LPT1OSC = OFF
#pragma config CCP2MX = ON
#pragma config STVREN = OFF
#pragma config ICPRT = OFF

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Definição e inicialização das variáveis Globais          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.

unsigned char leds_temp = 0;
unsigned char chaves_temp = 0;
unsigned int i = 0;
unsigned char unidade;
unsigned char dezena;
unsigned char centena;
unsigned char milhar;

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Constantes internas                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de constantes facilita a programação e a manutenção.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Declaração dos flags de software                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de flags ajuda na programação e economiza memória RAM.


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Definição e inicialização dos port's                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       PROTOTIPAGEM DE FUNÇÕES                         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void comando_lcd(unsigned char caracter);
void escreve_lcd(unsigned char caracter);
void limpa_lcd(void);
void inicializa_lcd(void);
void escreve_frase(const rom char *frase);
void escreve_pot( unsigned char dado );
unsigned char le_pot(void);

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              ENTRADAS                                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// As entradas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              SAÍDAS                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
```

```
//As saídas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define C_LEDS    LATAbits.LATA4   //PINO PARA ATIVAR BARRA DE LEDS
                                    //1 -> BARRA ATIVADA
                                    //0 -> BARRA DESATIVADA
#define CHAVES   PORTD

#define CS_POT   LATCbits.LATC0    //PINO CS DO POTENCIOMETRO DIGITAL

#define SCK_POT        LATAbits.LATA2    //PINO SCK DO POTENCIOMETRO DIGITAL

#define SDO_POT        LATAbits.LATA5    //PINO SDI DO POTENCIOMETRO DIGITAL

#define rs        PORTEbits.RE0    // via do lcd que sinaliza recepção
                                                  // de dados ou comando

#define enable    PORTEbits.RE1    // enable do lcd

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *          Rotina para escrita de dado serial (SPI por software)      *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void escreve_pot( unsigned char dado )
{
 unsigned char mascara;

 mascara = 0x80;

 SCK_POT = 1;

 do
 {
        SCK_POT = 0;

        if( dado & mascara )
               SDO_POT = 1;
        else
               SDO_POT = 0;

        SCK_POT = 1;

        mascara >>= 1;
 }
 while( mascara );

}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                Rotina que envia um COMANDO para o LCD                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void comando_lcd(unsigned char caracter)
{
    rs = 0;                         // seleciona o envio de um comando
    PORTD = caracter;               // carrega o PORTD com o caracter
    enable = 1 ;                    // gera pulso no enable
    Delay10TCYx(1);                 // espera 10 microsegundos
    enable = 0;                     // desce o pino de enable
    Delay10TCYx(4);                 // espera mínimo 40 microsegundos
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Rotina que envia um DADO a ser escrito no LCD             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
```

```
void escreve_lcd(unsigned char caracter)
{
    rs = 1;                     // seleciona o envio de um comando
    PORTD = caracter;           // carrega o PORTD com o caracter
    enable = 1;                 // gera pulso no enable
    Delay10TCYx(1);             // espera 10 microsegundos
    enable = 0;                 // desce o pino de enable
    Delay10TCYx(4);             // espera mínimo 40 microsegundos
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            Função para limpar o LCD                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void limpa_lcd(void)
{
    comando_lcd(0x01);            // limpa lcd
    Delay1KTCYx(2);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                 Inicialização do Display de LCD                         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void inicializa_lcd(void)
{
    comando_lcd(0x30);            // envia comando para inicializar display
    Delay1KTCYx(4);               // espera 4 milisengundos

    comando_lcd(0x30);            // envia comando para inicializar display
    Delay10TCYx(10);              // espera 100 microsengundos

    comando_lcd(0x30);            // envia comando para inicializar display

    comando_lcd(0x38);            // liga o display, sem cursor e sem blink

    limpa_lcd();                  // limpa lcd

    comando_lcd(0x0c);            // display sem cursor

    comando_lcd(0x06);            // desloca cursor para a direita
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Função para escrever uma frase no LCD                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void escreve_frase(const rom char *frase)
{
    do
    {
        escreve_lcd(*frase);
    }while(*++frase);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      FUNÇÃO PARA SEPARAR DÍGITOS                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void converte_bcd(unsigned char aux)
{
    unidade = 0;
    dezena = 0;
    centena = 0;
 milhar = 0;
    if (aux == 0)return;    //SE AUX. FOR IGUAL A ZERO, SAI
```

```
    while(aux--)
    {
        unidade++;
        if(unidade != 10)continue;
        unidade = 0;
        dezena++;
        if (dezena != 10)continue;
        dezena = 0;
        centena++;
        if (centena != 10)continue;
        centena = 0;
        milhar++;
     if (milhar != 10)continue;
        milhar = 0;
    }
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função Principal                             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void main ()
{
    PORTA = 0x00;               //Limpa PORTA
    PORTB = 0x00;               //Limpa PORTB
    PORTC = 0x00;               //Limpa PORTC
    PORTD = 0x00;               //Limpa PORTD
    PORTE = 0x00;               //Limpa PORTE

    LATA = 0x00;                //Limpa PORTA
    LATB = 0x00;                //Limpa PORTB
    LATC = 0x06;                //Liga as saídas para tensão de referência
    LATD = 0x00;                //Limpa PORTD
    LATE = 0x00;                //Limpa PORTE

    TRISA = 0b11001011;                 //CONFIG DIREÇÃO DOS PINOS PORTA
    TRISB = 0b00001111;                 //CONFIG DIREÇÃO DOS PINOS PORTB
    TRISC = 0b10111000;                 //CONFIG DIREÇÃO DOS PINOS PORTC
    TRISD = 0b00000000;                 //CONFIG DIREÇÃO DOS PINOS PORTD
    TRISE = 0b00000100;                 //CONFIG DIREÇÃO DOS PINOS PORTE

 ADCON1 = 0b00001100;

 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Inicialização do Sistema                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
 Delay10KTCYx(10);

 OpenADC(ADC_FOSC_8 & ADC_LEFT_JUST & ADC_0_TAD, ADC_CH1 & ADC_INT_OFF &
ADC_VREFPLUS_VDD
            & ADC_VREFMINUS_VSS,12);

 inicializa_lcd();

 // Inicialização do potenciometro digital

 SCK_POT = 1;
 CS_POT = 1;

 Delay10KTCYx(10);

 CS_POT = 0;

 // Escrita em TCON, D8 = 1
 escreve_pot(0b01000001);
 // R1HW = 1, R1A = 1, R1W = 1, R1B = 1, R0HW = 1, R0A = 0, R0W = 1, R0B = 1
```

```
  escreve_pot(0b11111011);

  // Escrita em Wiper 0
  escreve_pot(0b00000000);
  // P0W = P0B (curto)  -> U3A torna-se seguidor de tensão
  escreve_pot(0b00000000);

  // Escrita em Wiper 1
  escreve_pot(0b00010000);
  // P1W = P1B (curto)  -> Terminal + de U3A igual à 2,5 Volts
  escreve_pot(0b00000000);

  CS_POT = 1;

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Rotina Principal                             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

    while(1)
    {
        C_LEDS = 0;         // Desabilita a a barra de leds associada ao
                            // aos pinos do PORTD

        TRISD = 0xFF;       // Configura PORTD do microcontrolador como
                            // entrada

        Delay1KTCYx(1);           // Atraso para acomodação do PORT

        leds_temp = PORTD;  // Realiza leitura de todas as chaves
                            // e armazena na variável auxiliar

        CS_POT = 0;

        // Escrita em Wiper 1
        escreve_pot(0b00010000);
        // P0W = valor binário das chaves (Rs = chaves/256), divisor entre 5 - 2,5 Volts
        escreve_pot(leds_temp);

        CS_POT = 1;

        Delay1KTCYx(1);           // Atraso para acomodação do PORT

        TRISD = 0x00;       // Configura PORTD do microcontrolador como
                            // saída

        Delay1KTCYx(1);             // Atraso para acomodação do PORT

        LATD = leds_temp;   // Realiza escrita no PORTD do valor
                            // correspondente as chaves acionadas

        C_LEDS = 1;         // Desabilita a a barra de leds associada ao
                            // aos pinos do PORTD

        Delay1KTCYx(1);           // Atraso para exibição dos valores das chaves
                            // através da barra de leds

        ConvertADC();           //Inicia conversão AD
        if(BusyADC())
          {
         converte_bcd((((unsigned int)ADRESH*50)/256));
           c omando_lcd(0xC6);              // posiciona o cursor na linha 2, coluna 7
            escreve_lcd (dezena + 0x30);
            escreve_lcd (',');
            escreve_lcd (unidade + 0x30);
            escreve_lcd ('V');
        }
```

```
  }
}Exemplo placa de experiência EXP04
```

Para a utilização dos exemplos USB, a disposição dos jumpers da McMaster deve ser a seguinte:

“CAP USB”	->	Posição “OFF”
“DATA I2C”	->	Posição “ABERTO”
“CLOCK I2C”	->	Posição “ABERTO”
“COLUNA 2 RB0”	->	Posição “ON”
“COLUNA 1 RB0”	->	Posição “ON”
“LINHA 1”	->	Posição “ON”
“LEDS ESPECIAIS”	->	Posição “ON”

27. Exemplo placa de experiência EXP04

27.1. Objetivo

O objetivo desta experiência é validar a placa de experiência EXP04.

27.2. Descrição

Esta placa de experiência não possui um código exemplo específico, pois é uma placa de prototipagem que permite acesso a todos os pinos do microcontrolador conectado ao McMaster.

Obs: (1) O usuário deve levar em consideração as características elétricas dos pinos do microcontrolador utilizado visando obter uma interface segura entre o seu protótipo e o McMaster. Consulte sempre o datasheet do dispositivo antes de efetuar qualquer tipo de interligação.

28. Exemplo placa de experiência EXP05

28.1. Objetivo

O objetivo desta experiência é validar a placa de experiência EXP05.

28.2. Descrição

A placa de experiência EXP05 oferece como periferia ao microcontrolador entradas e saídas optoacopladas, entradas e saídas analógicas e uma interface de comunicação padrão RS-485. Seus exemplos estão divididos da seguinte forma:

- Exemplo 5a - leitura das entradas optoacopladas e exibição de estados através do LCD
- Exemplo 5b - escrita nas saídas optoacopladas e exibição de estados através do LCD. Obs: Válido somente para o microcontrolador PIC18F4520
- Exemplo 5c – geração de sinais analógicos de acordo com o posicionamento do potênciometro do kit e subseqüente conversão e exibição dos valores através do LCD por intermédio das interfaces analógicas amplificada e 4-20mA. Obs: Válidos somente para o microcontrolador PIC18F4520
- Exemplo 5d – leitura de entradas e envio do valor pelo transmissor para um receptor através da interface RS-485.

28.3. Código

-

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Descrição geral                                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Exemplo de Código para utilização com a McMaster               *
 *                          EXP05A                                            *
 *                                                                            *
 *                          MOSAICO                                           *
 *                                                                            *
 *  TEL: (0XX11) 4992-8775              SITE: www.mosaico.com.br              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                             *
 *   DATA : 18/01/2010                                                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Descrição geral                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*
ESTE EXEMPLO FOI ELABORADO PARA DEMONSTRAR O FUNCIONAMENTO DAS ENTRADAS
OPTOACOPLADAS. CADA ENTRADA È EXIBIDA NO LCD ATRAVÉS DA INDICAÇÂO "1" PARA
TENSÂO DE ENTRADA IGUAL A 24V E "0" PARA TENSÃO DE ENTRADA IGUAL A 0V.
O PROCESSO DE ATUALIZAÇÃO DAS ENTRADAS É EFETUADA NUMA BASE DE TEMPO DE
0,25 SEGUNDOS SENDO QUE A CADA INSTANTE DE ATUALIZAÇÃO O LED RC0 É INVERTIDO
DE ESTADO.
*/
/*      CONFIGURAÇÂO DOS JUMPERS DA PLACA:

        CAP USB(RC3)                            -> ON
        DATA I2C                                      -> OFF
        CLOCK I2C                                     -> OFF
        COLUNA 2(RB1)                           -> ON
        COLUNA 1(RB0)                           -> ON
        LINHA 1 / DISPLAY MILHAR(RB4)    -> ON
        LEDS ESPECIAS (RC0,RC1,RC2)             -> ON

*/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          DEFINIÇÃO PIC                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <p18F4550.h>            // Register definitions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               INCLUDES DAS FUNÇÕES DE PERIFÉRICOS DO PIC                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <pwm.h>                //PWM library functions
#include <adc.h>                //ADC library functions
#include <timers.h>             //Timer library functions
#include <delays.h>             //Delay library functions
#include <i2c.h>                //I2C library functions
#include <stdlib.h>             //Library functions
#include <usart.h>              //USART library functions


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      Configurações para gravação                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
```

```
#pragma config FOSC = XT_XT
#pragma config CPUDIV = OSC1_PLL2
#pragma config WDT = OFF
#pragma config WDTPS = 128
#pragma config LVP = OFF
#pragma config PWRT = ON
#pragma config BOR = OFF
#pragma config BORV = 0
#pragma config PBADEN = OFF
#pragma config DEBUG = OFF
#pragma config PLLDIV = 1
#pragma config USBDIV = 1
#pragma config FCMEN = OFF
#pragma config IESO = OFF
#pragma config VREGEN = OFF
#pragma config MCLRE = ON
#pragma config LPT1OSC = OFF
#pragma config CCP2MX = ON
#pragma config STVREN = OFF
#pragma config ICPRT = OFF
#pragma config XINST = OFF

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Definição e inicialização das variáveis Globais          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.
unsigned char contador_varredura;           // Variável mutiplicadora do tempo
                                            // de varredura

unsigned char valor_entradas;               // Variável para armazenamento
                                            // dos estados da entrada

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Constantes internas                         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de constantes facilita a programação e a manutenção.

#define T_VARREDURA          5              // 5x50ms = 0,25 segundo para
                                            // varredura das entradas

#define SAIDA               0x00            // Transforma barramento em saída
#define ENTRADA             0xFF            // Transforma barramento em entrada

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Declaração dos flags de software                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de flags ajuda na programação e economiza memória RAM.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Definição e inicialização dos port's                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       PROTOTIPAGEM DE FUNÇÕES                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void HIGH_int(void);
void isr_high(void);
void timer2_isr( void );
void inicializa_timer(void);

void comando_lcd(unsigned char caracter);
void escreve_lcd(unsigned char caracter);
void limpa_lcd(void);
void inicializa_lcd(void);
```

```
void escreve_frase(const rom char *frase);
void tela_principal(void);

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              ENTRADAS                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// As entradas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.
#define     BUS_ENTRADAS    PORTD       // Para leitura dos estados da entrada

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                               SAÍDAS                                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//As saídas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define rs          PORTEbits.RE0    // via do lcd que sinaliza recepção de dados ou
comando
#define enable      PORTEbits.RE1    // enable do lcd

#define CLK_ENTRADA     LATAbits.LATA5  // Sinal de clk do latch de entrada
#define LED_VARREDURA   LATCbits.LATC0  // Led de sinalização do instante de varredura

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Função de tratamento de interrupção de Timer2              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void timer2_isr( void )
{
    unsigned char mascara = 0x80;           // Bit mais significativo em 1
    unsigned char coluna_lcd = 0xC1;        // Linha 1 coluna 0

    contador_varredura--;
    if( !contador_varredura )       // Alcançamos o tempo de varredura?
    {
        // Sim. Recarregamos multiplicador e atualizamos LED_VARREDURA
     contador_varredura = T_VARREDURA;
     LED_VARREDURA = ~LED_VARREDURA;

        TRISD = ENTRADA;                    // Barramento como entrada
        Nop();
        Nop();
        Nop();

        CLK_ENTRADA = 1;                    // Borda de subida no pino CLK
        Nop();
        Nop();
        Nop();

        valor_entradas = BUS_ENTRADAS;      // Obtém valores das entradas

        Nop();
        Nop();
        Nop();
        CLK_ENTRADA = 0;

        Nop();
        Nop();
        Nop();
        TRISD = SAIDA;                      // Barramento como saida


        while( mascara )
        {
            comando_lcd(coluna_lcd);    // posiciona LCD de acordo com a entrada lida
```

```
            // Teste bit a bit para determinação da entrada ativa
            if( mascara & valor_entradas )
            {
                escreve_lcd('1');
            }
            else
            {
                escreve_lcd('0');
            }

            mascara >>= 1;                  // Rotaciona um bit para esquerda
            coluna_lcd = coluna_lcd + 2;    // Avança duas colunas no LCD
        }

    }

    PIR1bits.TMR2IF  = 0;           // Limpa flag de interrupção do Timer2
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                Rotina que envia um COMANDO para o LCD                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void comando_lcd(unsigned char caracter)
{
    rs = 0;                         // seleciona o envio de um comando
    PORTD = caracter;               // carrega o PORTD com o caracter
    enable = 1 ;                    // gera pulso no enable
    Delay10TCYx(1);                 // espera 10 microsegundos
    enable = 0;                     // desce o pino de enable
    Delay10TCYx(4);                 // espera mínimo 40 microsegundos
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *            Rotina que envia um DADO a ser escrito no LCD                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void escreve_lcd(unsigned char caracter)
{
    rs = 1;                     // seleciona o envio de um comando
    PORTD = caracter;           // carrega o PORTD com o caracter
    enable = 1;                 // gera pulso no enable
    Delay10TCYx(1);             // espera 10 microsegundos
    enable = 0;                 // desce o pino de enable
    Delay10TCYx(4);             // espera mínimo 40 microsegundos
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                             Função para limpar o LCD                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void limpa_lcd(void)
{
    comando_lcd(0x01);            // limpa lcd
    Delay1KTCYx(2);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Inicialização do Display de LCD                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void inicializa_lcd(void)
{
    Delay1KTCYx(15);            // espera 15 milisengundos
    comando_lcd(0x30);          // envia comando para inicializar display
    Delay1KTCYx(4);             // espera 4 milisengundos
```

```
    comando_lcd(0x30);          // envia comando para inicializar display
    Delay10TCYx(10);            // espera 100 microsengundos

    comando_lcd(0x30);          // envia comando para inicializar display

    comando_lcd(0x38);          // liga o display, sem cursor e sem blink

    limpa_lcd();                // limpa lcd

    comando_lcd(0x0c);          // display sem cursor

    comando_lcd(0x06);          // desloca cursor para a direita
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Função para escrever uma frase no LCD                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void escreve_frase(const rom char *frase)
{
    do
    {
        escreve_lcd(*frase);
    }while(*++frase);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                 Inicialização do Timer 2                                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void inicializa_timer(void)
{
    /*** Timer2 - ~50ms: base de tempo para varredura das entradas ***/

    T2CON = 0x73;               // Timer2 parado, prescale 1:16, postcale
                                // 1:16, fonte de clock interno
    TMR2  = 0x00;               // Zera contador
    PR2   = 195;                // Carrega período do Timer2

    PIR1bits.TMR2IF  = 0;       // Limpa flag de interrupção do Timer2
    PIE1bits.TMR2IE  = 1;       // Habilita interrupção Timer2

    T2CONbits.TMR2ON = 1;       // Liga Timer2
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Tela Principal                                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void tela_principal(void)
{
    comando_lcd(0x80);            // posiciona o cursor na linha 0, coluna 0
    escreve_frase("I8I7I6I5I4I3I2I1");
    comando_lcd(0xC0);            // posiciona o cursor na linha 1, coluna 0
    escreve_frase(" 0 0 0 0 0 0 0 0");
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função Principal                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void main()
{
    PORTA = 0x00;               //Limpa PORTA
    PORTB = 0x00;               //Limpa PORTB
    PORTC = 0x00;               //Limpa PORTC
```

```
    PORTD = 0x00;               //Limpa PORTD
    PORTE = 0x00;               //Limpa PORTE

    LATA = 0x00;                //Limpa PORTA
    LATB = 0x00;                //Limpa PORTB
    LATC = 0x00;                //Limpa PORTC
    LATD = 0x00;                //Limpa PORTD
    LATE = 0x00;                //Limpa PORTE

    TRISA = 0b11001111;                 //CONFIG DIREÇÃO DOS PINOS PORTA0
    TRISB = 0b00001111;                 //CONFIG DIREÇÃO DOS PINOS PORTB
    TRISC = 0b11111110;                 //CONFIG DIREÇÃO DOS PINOS PORTC
    TRISD = 0b00000000;                 //CONFIG DIREÇÃO DOS PINOS PORTD
    TRISE = 0b00000100;                 //CONFIG DIREÇÃO DOS PINOS PORTE

    ADCON1 = 0b00001111;                //DESLIGA CONVERSORES A/D
    CMCON  = 0b00000111;

    INTCONbits.PEIE = 1;        // Habilita interrupção de periféricos
 INTCONbits.GIE  = 1;        // Habilita interrupção global

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Inicialização do Sistema                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
    inicializa_lcd();           // configura o lcd
    tela_principal();           // imprime a tela principal no LCD
    inicializa_timer();         // Inicializa Timer 2

    // Carrega valor referência para varredura
    contador_varredura = T_VARREDURA;

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                             Rotina Principal                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
    while(1)
    {
        ClrWdt();               //LIMPA O WDT
    }
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *            ROTINA DE TRATAMENTO DE INT DE ALTA PRIORIDADE             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#pragma code VETOR_INT_HIGH = 0x0008
void HIGH_int(void)
{
    _asm
    goto isr_high
    _endasm
}
#pragma code

#pragma interrupt isr_high
void isr_high(void)
{
    if( PIR1bits.TMR2IF && PIE1bits.TMR2IE )
    {
        timer2_isr();
    }
}
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                                Descrição geral                                                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                Exemplo de Código para utilização com a McMaster                                *
 *                                EXP05C                                                         *
 *                                                                                               *
 *                                MOSAICO                                                        *
 *                                                                                               *
 *  TEL: (0XX11) 4992-8775                    SITE: www.mosaico.com.br                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                                                *
 *   DATA : 18/01/2010                                                                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                                Descrição geral                                                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*
ESTE EXEMPLO FOI ELABORADO PARA DEMONSTRAR O FUNCIONAMENTO DAS SAÌDAS
DIGITAL-ANALÒGICO PWM AN4 e AN5 EM CONJUNTO AS ENTRADAS ANALÓGICO-DIGITAL
AMPLIFICADAS AN2 E AN3. AS SAÍDAS AN4 E AN5 GERAM, RESPECTIVAMENTE, 0-15mA
E 0-100mV DE FORMA A OBTER VALORES NAS ENTRADAS AN2 E AN3 DE ACORDO COM AS
SEGUINTES RELAÇÕES:
                              **ENTRADA AN2**

[ V(SAÍDA AN4) / (220 + 100) ] = I(ENTRADA AN2) = TENSÃO CONVERTIDA EM AN2 / 250

                              **ENTRADA AN3**

          V(SAÍDA AN5) = V(ENTRADA AN3)*50 = TENSÃO CONVERTIDA EM AN3


AS SAÍDAS AN4 E AN5 DEVEM SER INTERLIGADAS, ATRAVÉS DE JUMPERS, RESPECTIVAMENTE
AS ENTRADAS AN2 E AN3.

*/
/*      CONFIGURAÇÂO DOS JUMPERS DA PLACA:

        CAP USB(RC3)                                -> ON
        DATA I2C                                           -> OFF
        CLOCK I2C                                          -> OFF
        COLUNA 2(RB1)                               -> ON
        COLUNA 1(RB0)                               -> ON
        LINHA 1 / DISPLAY MILHAR(RB4)     -> ON
        LEDS ESPECIAS (RC0,RC1,RC2)                 -> ON

*/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                               DEFINIÇÃO PIC                                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <p18F4550.h>             // Register definitions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                INCLUDES DAS FUNÇÕES DE PERIFÉRICOS DO PIC                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <pwm.h>                //PWM library functions
#include <adc.h>                //ADC library functions
#include <timers.h>             //Timer library functions
```

```
#include <delays.h>             //Delay library functions
#include <i2c.h>                //I2C library functions
#include <stdlib.h>             //Library functions
#include <usart.h>              //USART library functions


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Configurações para gravação                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#pragma config FOSC = HS
#pragma config CPUDIV = OSC1_PLL2
#pragma config WDT = OFF
#pragma config WDTPS = 128
#pragma config LVP = OFF
#pragma config PWRT = ON
#pragma config BOR = OFF
#pragma config BORV = 0
#pragma config PBADEN = OFF
#pragma config DEBUG = OFF
#pragma config PLLDIV = 1
#pragma config USBDIV = 1
#pragma config FCMEN = OFF
#pragma config IESO = OFF
#pragma config VREGEN = OFF
#pragma config MCLRE = ON
#pragma config LPT1OSC = OFF
#pragma config CCP2MX = ON
#pragma config STVREN = OFF
#pragma config ICPRT = OFF
#pragma config XINST = OFF

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Definição e inicialização das variáveis Globais          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.
unsigned char   milhar, centena, dezena;

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Constantes internas                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de constantes facilita a programação e a manutenção.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                    Declaração dos flags de software                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de flags ajuda na programação e economiza memória RAM.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Definição e inicialização dos port's                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        PROTOTIPAGEM DE FUNÇÕES                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void HIGH_int(void);
void isr_high(void);
void timer2_isr( void );

void comando_lcd(unsigned char caracter);
void escreve_lcd(unsigned char caracter);
void limpa_lcd(void);
void inicializa_lcd(void);
void escreve_frase(const rom char *frase);
void tela_principal(void);
```

```
void converte_bcd(unsigned int aux);

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              ENTRADAS                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// As entradas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                               SAÍDAS                                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//As saídas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define rs        PORTEbits.RE0    // via do lcd que sinaliza recepção de dados ou
comando
#define enable    PORTEbits.RE1    // enable do lcd

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               Rotina que envia um COMANDO para o LCD                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void comando_lcd(unsigned char caracter)
{
    rs = 0;                         // seleciona o envio de um comando
    PORTD = caracter;               // carrega o PORTD com o caracter
    enable = 1 ;                    // gera pulso no enable
    Delay10TCYx(1);                 // espera 10 microsegundos
    enable = 0;                     // desce o pino de enable
    Delay10TCYx(4);                 // espera mínimo 40 microsegundos
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *            Rotina que envia um DADO a ser escrito no LCD                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void escreve_lcd(unsigned char caracter)
{
    rs = 1;                     // seleciona o envio de um comando
    PORTD = caracter;           // carrega o PORTD com o caracter
    enable = 1;                 // gera pulso no enable
    Delay10TCYx(1);             // espera 10 microsegundos
    enable = 0;                 // desce o pino de enable
    Delay10TCYx(4);             // espera mínimo 40 microsegundos
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            Função para limpar o LCD                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void limpa_lcd(void)
{
    comando_lcd(0x01);            // limpa lcd
    Delay1KTCYx(2);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                 Inicialização do Display de LCD                         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void inicializa_lcd(void)
{
    Delay1KTCYx(15);            // espera 15 milisengundos
    comando_lcd(0x30);          // envia comando para inicializar display
    Delay1KTCYx(4);             // espera 4 milisengundos
```

```
    comando_lcd(0x30);          // envia comando para inicializar display
    Delay10TCYx(10);            // espera 100 microsengundos

    comando_lcd(0x30);          // envia comando para inicializar display

    comando_lcd(0x38);          // liga o display, sem cursor e sem blink

    limpa_lcd();                // limpa lcd

    comando_lcd(0x0c);          // display sem cursor

    comando_lcd(0x06);          // desloca cursor para a direita
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Função para escrever uma frase no LCD                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void escreve_frase(const rom char *frase)
{
    do
    {
        escreve_lcd(*frase);
    }while(*++frase);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            Converte BCD                                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void converte_bcd(unsigned int aux)
{
    dezena  = 0;
    centena = 0;
    milhar  = 0;
    if (aux == 0)return;
    while(aux--)
    {
        dezena++;
        if(dezena != 10)continue;
        dezena = 0;
        centena++;
        if (centena != 10)continue;
        centena = 0;
        milhar++;
        if (milhar != 10)continue;
        milhar = 0;
    }
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Tela Principal                                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void tela_principal(void)
{
    comando_lcd(0x80);            // posiciona o cursor na linha 0, coluna 0
    escreve_frase("A4       A2      ");
    comando_lcd(0xC0);            // posiciona o cursor na linha 1, coluna 0
    escreve_frase("A5       A3      ");
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função Principal                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void main()
```

```
{
    PORTA = 0x00;               //Limpa PORTA
    PORTB = 0x00;               //Limpa PORTB
    PORTC = 0x00;               //Limpa PORTC
    PORTD = 0x00;               //Limpa PORTD
    PORTE = 0x00;               //Limpa PORTE

    LATA = 0x00;                //Limpa PORTA
    LATB = 0x00;                //Limpa PORTB
    LATC = 0x00;                //Limpa PORTC
    LATD = 0x00;                //Limpa PORTD
    LATE = 0x00;                //Limpa PORTE

    TRISA = 0b11101111;                 //CONFIG DIREÇÃO DOS PINOS PORTA0
    TRISB = 0b00001111;                 //CONFIG DIREÇÃO DOS PINOS PORTB
    TRISC = 0b10110000;                 //CONFIG DIREÇÃO DOS PINOS PORTC
    TRISD = 0b00000000;                 //CONFIG DIREÇÃO DOS PINOS PORTD
    TRISE = 0b00000100;                 //CONFIG DIREÇÃO DOS PINOS PORTE

    ADCON1 = 0b00001011;                //LIGA CONVERSORES A/D - AN0 a AN3
    CMCON  = 0b00000111;

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      Inicialização do Sistema                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
    inicializa_lcd();           // configura o lcd

    //CONFIGURAÇÃO DO AD
    OpenADC(ADC_FOSC_8 & ADC_RIGHT_JUST & ADC_0_TAD, ADC_CH0 & ADC_INT_OFF &
ADC_REF_VDD_VSS,ADC_4ANA );

        //CONFIGURAÇÃO DO TIMER2
        OpenTimer2(TIMER_INT_OFF & T2_PS_1_4 & T2_POST_1_10);

 // Configura Timer2 em 7KHz acarretando em 9 bits de resolução
 PR2 = 34;

    //CONFIGURAÇÃO DO MÓDULO CCP1 -> PWM
    CCP1CON = 0B00001100;

    //CONFIGURAÇÃO DO MÓDULO CCP2 -> PWM
    CCP2CON = 0B00001111;

    tela_principal();           // imprime a tela principal no LCD

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            Rotina Principal                         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
    while(1)
    {
        unsigned int aux;

        ClrWdt();               //LIMPA O WDT

        // Seleciona canal analógico do potênciometro
        SetChanADC( ADC_CH0 );
        ConvertADC();                   // Realiza uma conversão
        while(BusyADC());           // Verifica se o conversor finalizou conversão

        // Atualiza AN4 com fundo de escala em 71 unidades PWM
        // Para exibição no LCD fundo de escala em 15mA
        aux = (unsigned int)(ReadADC()*(137/1023.0) );
        // A corrente máxima de saída é 15mA...
        SetDCPWM2( aux );
        converte_bcd( (unsigned int)(aux*(150/137.0)) );
```

```
            comando_lcd(0x84);              // posiciona o cursor na linha 0, coluna 4
            escreve_lcd( (milhar+0x30) );
            escreve_lcd( (centena+0x30) );
            escreve_frase("mA");

            // Atualiza AN5 com fundo de escala em 71 unidades PWM
            // Para exibição no LCD fundo de escala em 100mV
            aux = (unsigned int)(ReadADC()*(137/1023.0) );
            // Limita saída em 100mV para não saturar a entrada AN3
            SetDCPWM1( (unsigned int)(aux*(3/137.0)) );
            converte_bcd( (unsigned int)(aux*(100/137.0)) );

            comando_lcd(0xC3);              // posiciona o cursor na linha 2, coluna 3
            escreve_lcd( (milhar+0x30) );
            escreve_lcd( (centena+0x30) );
            escreve_lcd( (dezena+0x30) );
            escreve_frase("mV");

         // Seleciona canal analógico AN2 ( Conversor 4 a 20 mA )
         SetChanADC( ADC_CH2 );
            ConvertADC();                   // Realiza uma conversão
            while(BusyADC());           // Verifica se o conversor finalizou conversão

            // Fundo de escala em 5V
            aux = (unsigned int)((ReadADC()/1023.0)*50);
            converte_bcd( aux );

            comando_lcd(0x8C);              // posiciona o cursor na linha 1, coluna 13
            escreve_lcd( (centena+0x30) );
            escreve_lcd('.');
            escreve_lcd( (dezena+0x30) );
            escreve_frase("V");

            // Seleciona canal analógico AN3 ( Conversor com ganho 50 )
         SetChanADC( ADC_CH3 );
            ConvertADC();               // Realiza uma conversão
            while(BusyADC());   // Verifica se o conversor finalizou conversão

            // Fundo de escala em 5V
            aux = (unsigned int)((ReadADC()/1023.0)*50);
            converte_bcd( aux );

            comando_lcd(0xCC);              // posiciona o cursor na linha 2, coluna 13
            escreve_lcd( (centena+0x30) );
            escreve_lcd('.');
            escreve_lcd( (dezena+0x30) );
            escreve_frase("V");
      }
}
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Descrição geral                                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Exemplo de Código para utilização com a McMaster               *
 *                              EXP05D                                        *
 *                                                                            *
 *                              MOSAICO                                       *
 *                                                                            *
 *  TEL: (0XX11) 4992-8775                SITE: www.mosaico.com.br            *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                             *
 *   DATA : 18/01/2010                                                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Descrição geral                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*
ESTE EXEMPLO FOI ELABORADO PARA DEMONSTRAR O FUNCIONAMENTO DA INTERFACE
DE COMUNICAÇÃO SERIAL RS-485. O VALOR HEXADECIMAL CORRESPONDENTE AS ENTRADAS
DO KIT "TRANSMISSOR" SERÁ EXIBIDO NO LCD DO KIT "RECEPTOR", AMBOS INTERLIGADOS
ATRAVÉS DA INTERFACE. A VARREDURA E TRANSMISSÃO DO VALOR CORRESPONDENTE A CADA
ENTRADA DO KIT "TRANSMISSOR" SÃO EFETUADAS A CADA 100ms E A RECEPÇÃO DO DADO
POR PARTE DO KIT "RECEPTOR" NA ROTINA PRINCIPAL DO PROGRAMA. A VELOCIDADE DA
COMUNICAÇÃO È 1200BPS.

UTILIZE A DIRETIVA MODO_OPERACAO PARA COMPILAR O CÒDIGO NA VERSÃO TRANSMISSOR
OU RECEPTOR

*/
/*      CONFIGURAÇÂO DOS JUMPERS DA PLACA:

        CAP USB(RC3)                              -> ON
        DATA I2C                                        -> OFF
        CLOCK I2C                                       -> OFF
        COLUNA 2(RB1)                             -> ON
        COLUNA 1(RB0)                             -> ON
        LINHA 1 / DISPLAY MILHAR(RB4)    -> ON
        LEDS ESPECIAS (RC0,RC1,RC2)               -> ON

*/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           DEFINIÇÃO PIC                                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <p18F4550.h>             // Register definitions

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               INCLUDES DAS FUNÇÕES DE PERIFÉRICOS DO PIC                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <delays.h>             //Delay library functions
#include <usart.h>              //USART library functions


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Configurações para gravação                         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#pragma config FOSC = XT_XT
#pragma config CPUDIV = OSC1_PLL2
```

```
#pragma config WDT = OFF
#pragma config WDTPS = 128
#pragma config LVP = OFF
#pragma config PWRT = ON
#pragma config BOR = OFF
#pragma config BORV = 0
#pragma config PBADEN = OFF
#pragma config DEBUG = OFF
#pragma config PLLDIV = 1
#pragma config USBDIV = 1
#pragma config FCMEN = OFF
#pragma config IESO = OFF
#pragma config VREGEN = OFF
#pragma config MCLRE = ON
#pragma config LPT1OSC = OFF
#pragma config CCP2MX = ON
#pragma config STVREN = OFF
#pragma config ICPRT = OFF
#pragma config XINST = OFF

#define TRANSMISSOR         0
#define RECEPTOR            1

#define MODO_OPERACAO       TRANSMISSOR

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Definição e inicialização das variáveis Globais          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Neste bloco estão definidas as variáveis globais do programa.

unsigned char contador_varredura;           // Variável mutiplicadora do tempo
                                            // de varredura

unsigned char valor_entradas;               // Variável para armazenamento
                                            // dos estados da entrada

#if( MODO_OPERACAO == RECEPTOR )
    unsigned char hexa_high, hexa_low;
#endif

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Constantes internas                         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de constantes facilita a programação e a manutenção.

#if( MODO_OPERACAO == TRANSMISSOR )
    #define T_VARREDURA    20           // 20x5ms = 100ms para
                                        // varredura das entradas
    #define ENTRADA        0xFF         // Transforma barramento em entrada
#else
    #define T_VARREDURA     5           // 5x5ms = 25ms para
                                        // varredura das entradas
#endif

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Declaração dos flags de software                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//A definição de flags ajuda na programação e economiza memória RAM.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Definição e inicialização dos port's                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      PROTOTIPAGEM DE FUNÇÕES                         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
```

```
#if( MODO_OPERACAO == TRANSMISSOR )
    void HIGH_int(void);
    void isr_high(void);
    void timer2_isr(void);
    void inicializa_timer(void);
#else
    void converte_hexadecimal(unsigned char aux);
#endif

void comando_lcd(unsigned char caracter);
void escreve_lcd(unsigned char caracter);
void limpa_lcd(void);
void inicializa_lcd(void);
void escreve_frase(const rom char *frase);
void tela_principal(void);

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              ENTRADAS                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// As entradas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#if( MODO_OPERACAO == TRANSMISSOR )
    #define     BUS_ENTRADAS    PORTD       // Para leitura dos estados da entrada
#endif

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                               SAÍDAS                                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//As saídas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define rs          PORTEbits.RE0    // via do lcd que sinaliza recepção de dados ou
comando
#define enable      PORTEbits.RE1    // enable do lcd

#if( MODO_OPERACAO == TRANSMISSOR )
    #define CLK_ENTRADA     LATAbits.LATA5  // Sinal de clk do latch de entrada
#else
    #define DIR             LATCbits.LATC0  // Pino de direcionamento do fluxo de dados
                                            // na interface RS-485
#endif

#define LED_VARREDURA   LATCbits.LATC1  // Led de sinalização do instante de varredura

#if( MODO_OPERACAO == TRANSMISSOR )
    /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
     *             ROTINA DE TRATAMENTO DE INT DE ALTA PRIORIDADE             *
     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

    #pragma code VETOR_INT_HIGH = 0x0008
    void HIGH_int(void)
    {
        _asm
        goto isr_high
        _endasm
    }
    #pragma code

    #pragma interrupt isr_high
    void isr_high(void)
    {
        if( PIR1bits.TMR2IF && PIE1bits.TMR2IE )
        {
            timer2_isr();
        }
```

```
    }
#endif

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Função de tratamento de interrupção de Timer2              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#if( MODO_OPERACAO == TRANSMISSOR )
    void timer2_isr( void )
    {
        contador_varredura--;
        if( !contador_varredura )       // Alcançamos o tempo de varredura?
        {
                // Sim. Recarregamos multiplicador e atualizamos LED_VARREDURA
                   contador_varredura = T_VARREDURA;
              LED_VARREDURA = ~LED_VARREDURA;


                    CLK_ENTRADA = 1;                    // Borda de subida no pino CLK
                    Nop();
                    Nop();
                    Nop();

                    valor_entradas = BUS_ENTRADAS;      // Obtém valores das entradas

                    Nop();
                    Nop();
                    Nop();
                    CLK_ENTRADA = 0;

                    while(BusyUSART());
                    WriteUSART( valor_entradas );
        }
        PIR1bits.TMR2IF  = 0;           // Limpa flag de interrupção do Timer2
    }
#endif

#if( MODO_OPERACAO == RECEPTOR )
    /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
     *                          Converte hexadecimal                      *
     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

    void converte_hexadecimal(unsigned char aux)
    {
        hexa_high = (aux >> 4) & 0b00001111;
        if( hexa_high > 9 )
            hexa_high += 55;
        else
            hexa_high += 48;

        hexa_low = aux & 0b00001111;
        if( hexa_low > 9 )
            hexa_low += 55;
        else
            hexa_low += 48;
    }
#endif

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               Rotina que envia um COMANDO para o LCD                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void comando_lcd(unsigned char caracter)
{
    rs = 0;                         // seleciona o envio de um comando
    PORTD = caracter;               // carrega o PORTD com o caracter
    enable = 1 ;                    // gera pulso no enable
```

```
    Delay10TCYx(1);                 // espera 10 microsegundos
    enable = 0;                     // desce o pino de enable
    Delay10TCYx(4);                 // espera mínimo 40 microsegundos
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *            Rotina que envia um DADO a ser escrito no LCD              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void escreve_lcd(unsigned char caracter)
{
    rs = 1;                     // seleciona o envio de um comando
    PORTD = caracter;           // carrega o PORTD com o caracter
    enable = 1;                 // gera pulso no enable
    Delay10TCYx(1);             // espera 10 microsegundos
    enable = 0;                 // desce o pino de enable
    Delay10TCYx(4);             // espera mínimo 40 microsegundos
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função para limpar o LCD                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void limpa_lcd(void)
{
    comando_lcd(0x01);            // limpa lcd
    Delay1KTCYx(2);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Inicialização do Display de LCD                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void inicializa_lcd(void)
{
    Delay1KTCYx(15);            // espera 15 milisengundos
    comando_lcd(0x30);          // envia comando para inicializar display
    Delay1KTCYx(4);             // espera 4 milisengundos

    comando_lcd(0x30);          // envia comando para inicializar display
    Delay10TCYx(10);            // espera 100 microsengundos

    comando_lcd(0x30);          // envia comando para inicializar display

    comando_lcd(0x38);          // liga o display, sem cursor e sem blink

    limpa_lcd();                // limpa lcd

    comando_lcd(0x0c);          // display sem cursor

    comando_lcd(0x06);          // desloca cursor para a direita
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *            Função para escrever uma frase no LCD                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void escreve_frase(const rom char *frase)
{
    do
    {
        escreve_lcd(*frase);
    }while(*++frase);
}

#if ( MODO_OPERACAO == TRANSMISSOR )
```

```
    /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
     *                  Inicialização do Timer 2                               *
     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

    void inicializa_timer( void )
    {
        /*** Timer2 - ~5ms: base de tempo para varredura das entradas ***/

        T2CON = 0x71;               // Timer2 parado, prescale 1:4, postcale
                                    // 1:16, fonte de clock interno
        TMR2  = 0x00;               // Zera contador
        PR2   = 78;                 // Carrega período do Timer2

        PIR1bits.TMR2IF  = 0;       // Limpa flag de interrupção do Timer2
        PIE1bits.TMR2IE  = 1;       // Habilita interrupção Timer2

        T2CONbits.TMR2ON = 1;       // Liga Timer2
    }
#endif

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Tela Principal                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void tela_principal(void)
{
    comando_lcd(0x80);            // posiciona o cursor na linha 0, coluna 0

    #if( MODO_OPERACAO == TRANSMISSOR )
        escreve_frase("   TRANSMISSOR  ");
    #else
        escreve_frase("    RECEPTOR    ");
    #endif

    comando_lcd(0xC0);            // posiciona o cursor na linha 1, coluna 0
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função Principal                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void main()
{
    PORTA = 0x00;               //Limpa PORTA
    PORTB = 0x00;               //Limpa PORTB
    PORTC = 0x00;               //Limpa PORTC
    PORTD = 0x00;               //Limpa PORTD
    PORTE = 0x00;               //Limpa PORTE

    LATA = 0x00;                //Limpa PORTA
    LATB = 0x00;                //Limpa PORTB
    LATC = 0x00;                //Limpa PORTC
    LATD = 0x00;                //Limpa PORTD
    LATE = 0x00;                //Limpa PORTE

    TRISA = 0b11001111;                 //CONFIG DIREÇÃO DOS PINOS PORTA0
    TRISB = 0b00001111;                 //CONFIG DIREÇÃO DOS PINOS PORTB
    TRISC = 0b11111100;                 //CONFIG DIREÇÃO DOS PINOS PORTC
    TRISD = 0b00000000;                 //CONFIG DIREÇÃO DOS PINOS PORTD
    TRISE = 0b00000100;                 //CONFIG DIREÇÃO DOS PINOS PORTE

    ADCON1 = 0b00001111;                //DESLIGA CONVERSORES A/D
    CMCON  = 0b00000111;

    INTCONbits.PEIE = 1;        // Habilita interrupção de periféricos
  INTCONbits.GIE  = 1;        // Habilita interrupção global
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Inicialização do Sistema                            *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
    inicializa_lcd();           // configura o lcd
    tela_principal();           // imprime a tela principal no LCD

    //CONFIGURAÇÃO DA USART
    OpenUSART(USART_TX_INT_OFF & USART_RX_INT_OFF & USART_ASYNCH_MODE &
              USART_EIGHT_BIT  & USART_CONT_RX    & USART_BRGH_HIGH,25);

    #if( MODO_OPERACAO == TRANSMISSOR )
        TRISD = ENTRADA;            // Barramento como entrada
        inicializa_timer();         // Inicializa Timer 2
    #else
        DIR = 0;                    // Modo recepção
    #endif

    // Carrega valor referência para varredura
    contador_varredura = T_VARREDURA;

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            Rotina Principal                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
    while(1)
    {
        ClrWdt();               //LIMPA O WDT

        #if( MODO_OPERACAO == RECEPTOR )
            if( DataRdyUSART() )
            {
                converte_hexadecimal( ReadUSART() );

                comando_lcd(0xC0);            // posiciona o cursor na linha 1, coluna
0
                escreve_frase("0x");
                escreve_lcd(hexa_high);
                escreve_lcd(hexa_low);
            }
        #endif
    }
}
```

29. Software de comunicação serial

O software disponível no CD pode ser utilizado para comunicação entre o PC e o McMaster. O software é útil para testar as experiências de número 17 e 20.

O software foi desenvolvido sob a plataforma Windows é deve ser instalado através do arquivo setup.exe. Caso seja necessário atualizar alguns arquivos do sistema permita que o próprio software execute esta tarefa; o micro será reiniciado e o arquivo setup.exe deverá ser executado novamente pelo usuário.

Para a utilização do software deve-se inicialmente selecionar a porta COM de comunicação e o baud rate desejado.

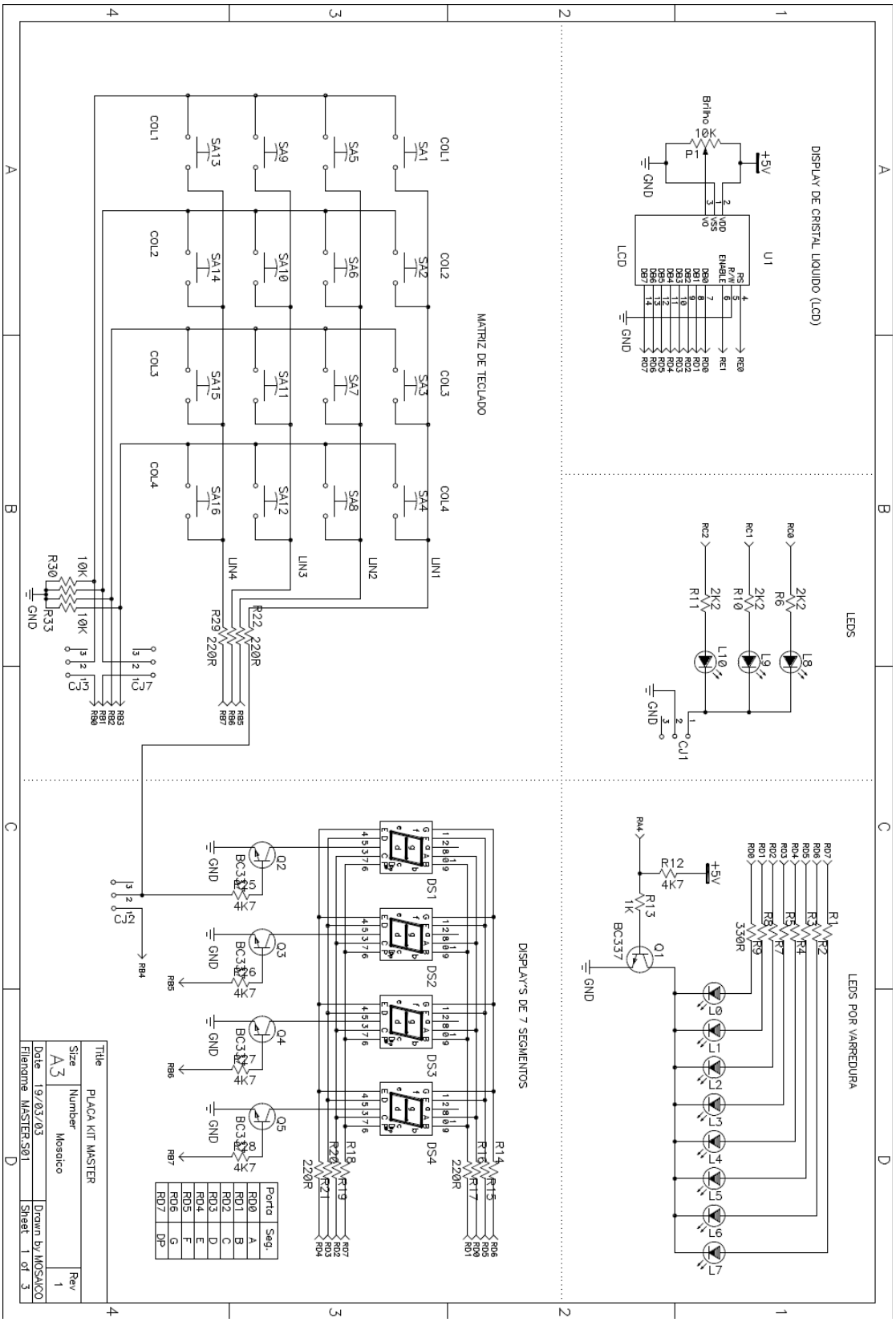
30. Software demo para teste do hardware

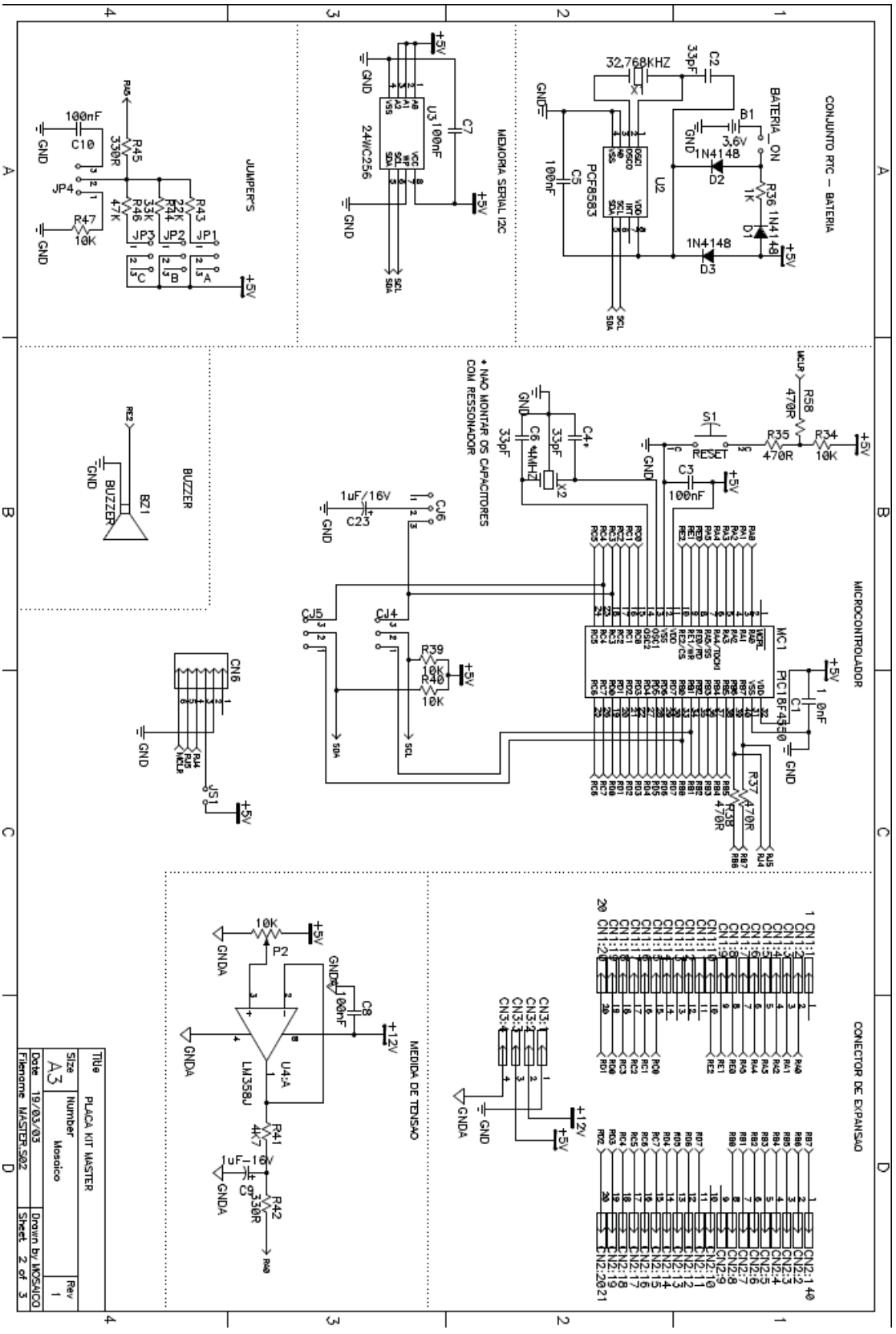
A fim de validar o hardware do McMaster, servindo como uma giga de testes, é fornecido também um software que pode ser utilizado para testar a funcionabilidade de quase todos os recursos do sistema. Para este software, não é fornecido o código fonte apenas o arquivo .HEX está disponível no CD. Como padrão este software já vem gravado no microcontrolador, porém a qualquer momento o usuário pode testar o funcionamento do hardware do McMaster regravando o arquivo .HEX. O software de teste pode ser executado sem interação com o usuário, porém recomendamos que o usuário faça a interação com o software a fim comprovar o correto funcionamento de todos os componentes do sistema.

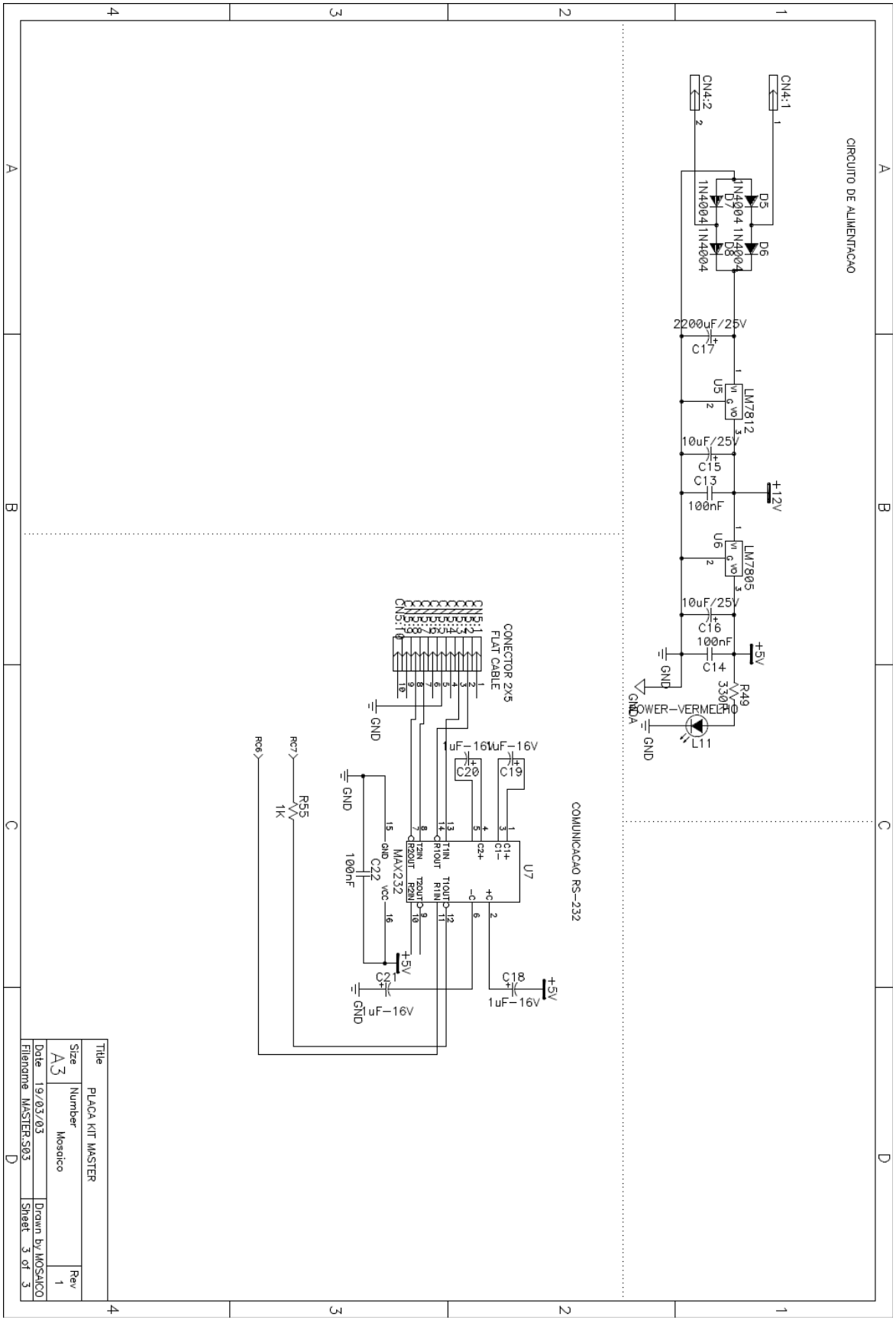
Obs: (1) O software demo utiliza a placa EXP01 para validação do hardware.

(2) O software demo do PIC18F4550 não aciona a lâmpada da placa EXP01 pois esta saída está reservada para utilização da comunicação USB

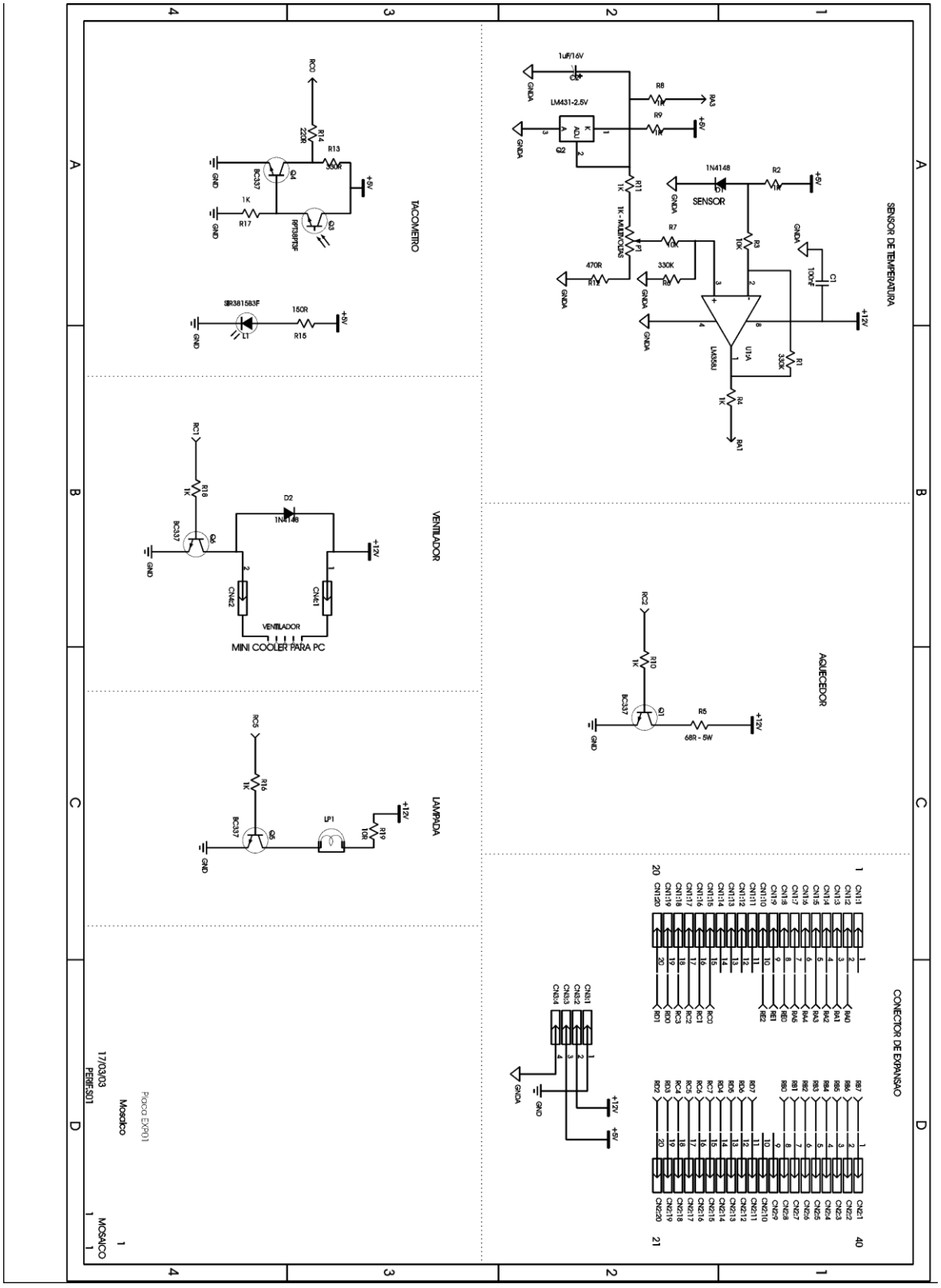
31. Apêndice A – Esquema elétrico completo do McMaster



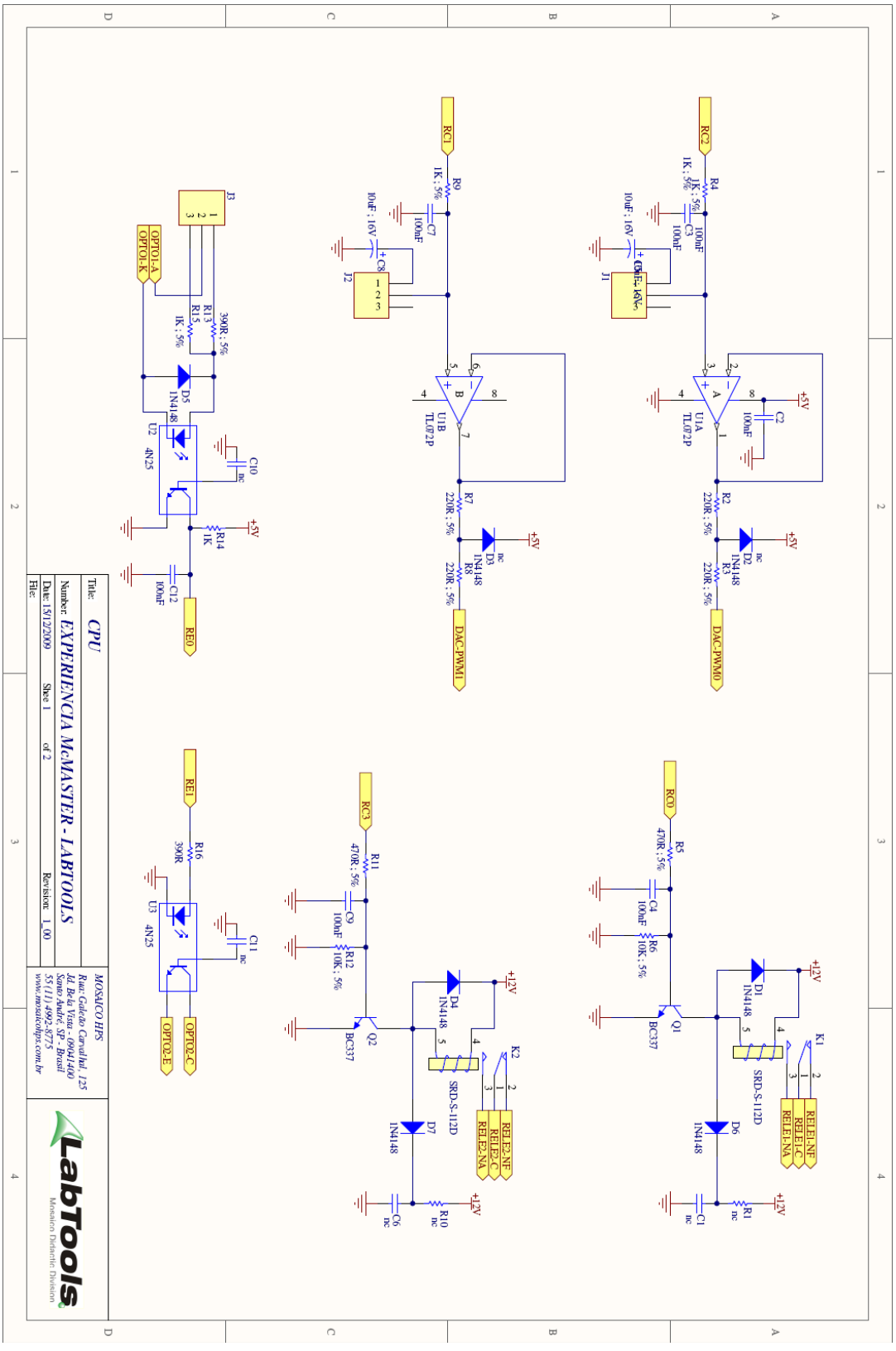

CONJUNTO RTC – BATERIA
MEMORIA SERIAL I2C
JUMPER'S
BUZZER
MICROCONTROLADOR
* NAO MONTAR OS CAPACITORES COM RESSONADOR
CONECTOR DE EXPANSAO
MEDIDA DE TENSAO
PLACA KIT MASTER
Mosaico
Drawn by MOSAICO
Sheet 2 of 3

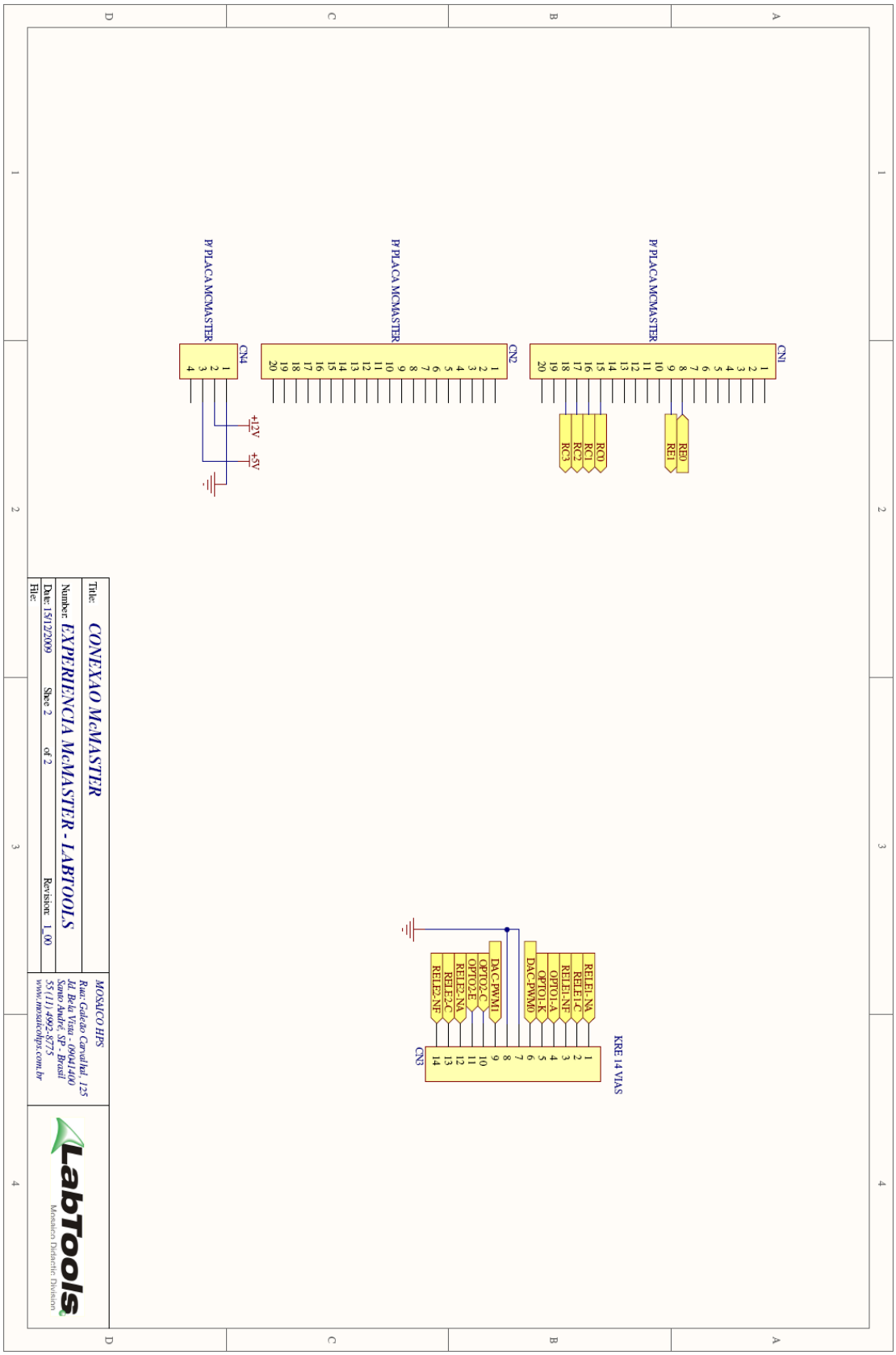

CIRCUITO DE ALIMENTACAO
CN4:1
CN4:2
D5
D6
D7
D8
1N4004
2200uF/25V
C17
U5
LM7812
10uF/25V
C15
C13
100nF
+12V
U6
LM7805
C16
C14
+5V
GND
R49
330R
POWER-VERMELHO
L11
CONECTOR 2X5
FLAT CABLE
CN5:1
CN5:10
COMUNICACAO RS-232
C20
C19
1uF-16V
U7
MAX232
C22
100nF
C21
C18
R55
1K
RC6
RC7
Title
PLACA KIT MASTER
Size
A3
Number
Mosaico
Rev
1
Date
19/03/03
Drawn by MOSAICO
Filename MASTER.SCH
Sheet 3 of 3

32. Apêndice B – Esquema elétrico completo da placa de experiências EXP01

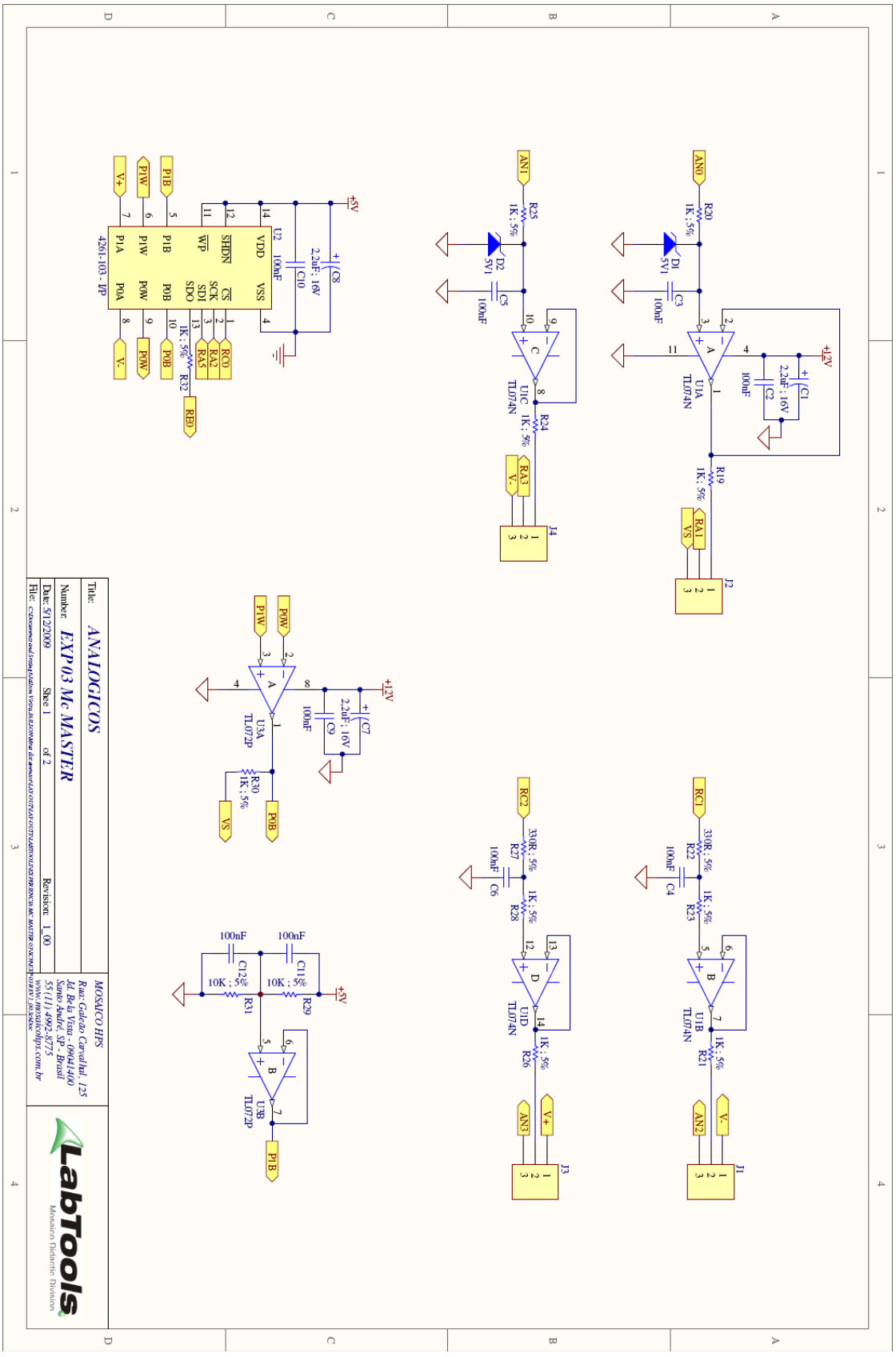


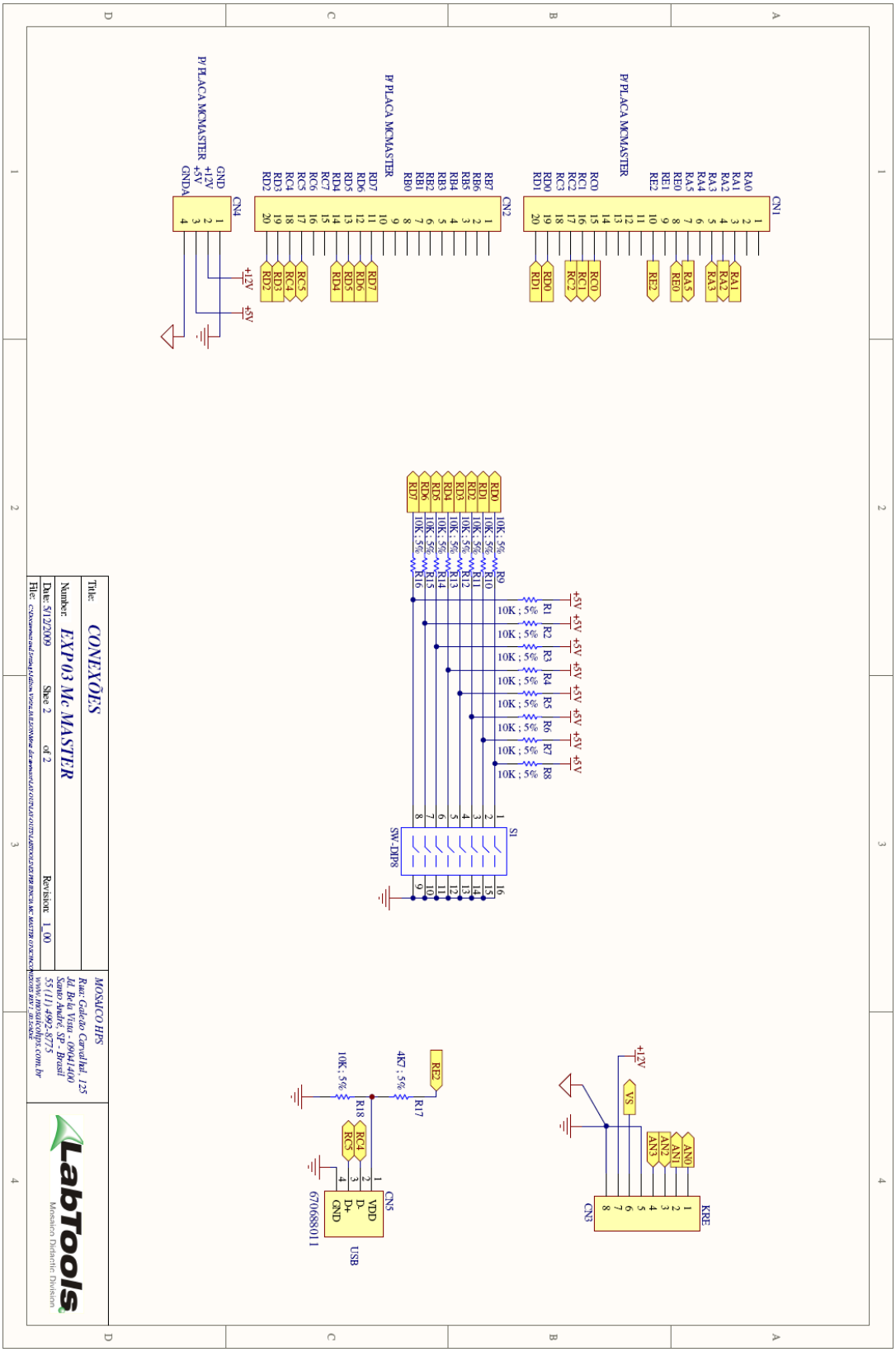
33. Apêndice B – Esquema elétrico completo da placa de experiências EXP02



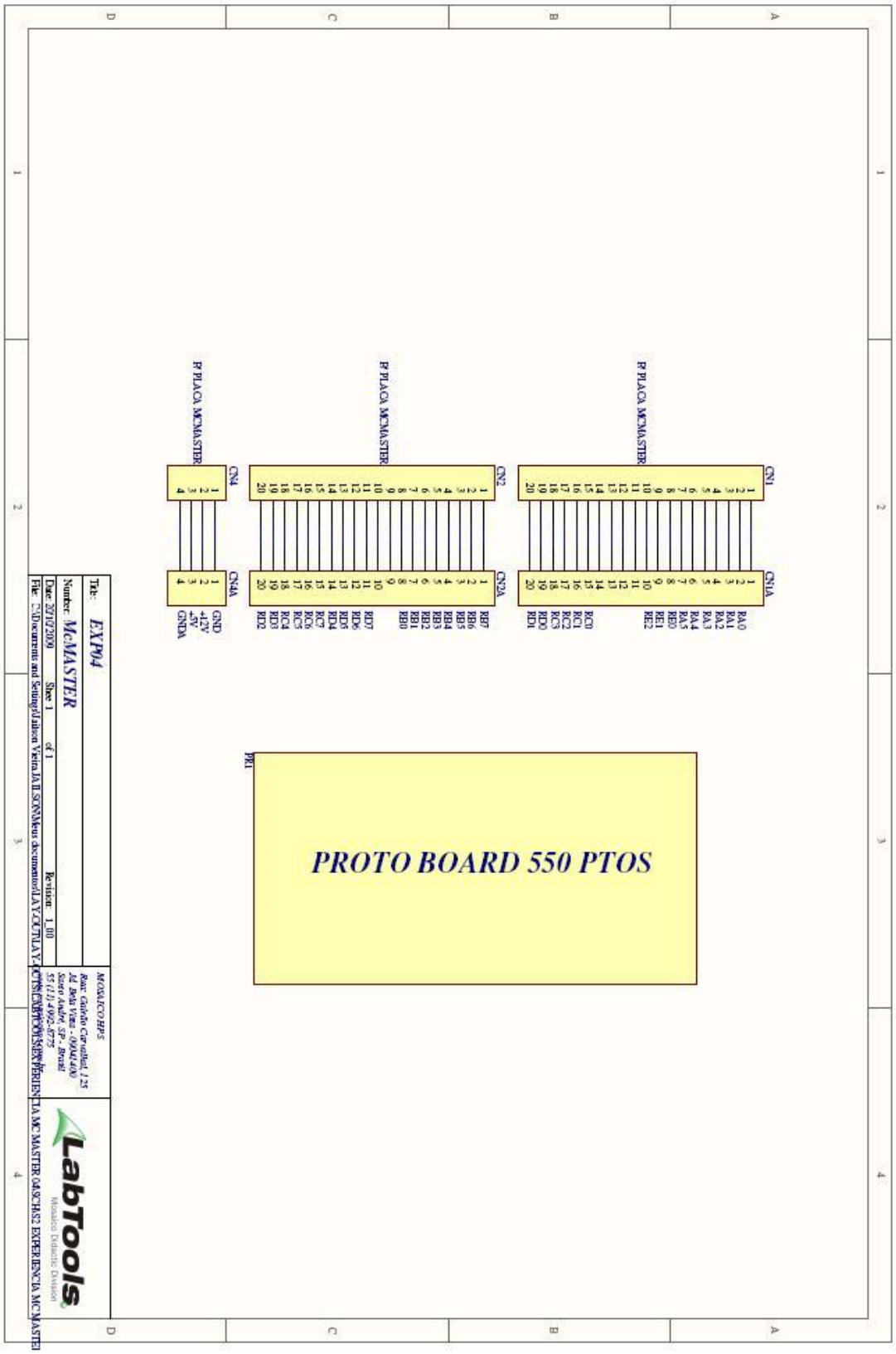

P/ PLACA MCMASTER
CN1
RC0
RC1
RC2
RC3
RE0
RE1
P/ PLACA MCMASTER
CN2
P/ PLACA MCMASTER
CN4
+12V
+5V
KRE 14 VIAS
CN3
RELE1-NA
RELE1-C
RELE1-NF
OPTO1-A
OPTO1-K
DAC-PWM0
DAC-PWM1
OPTO2-C
OPTO2-E
RELE2-NA
RELE2-C
RELE2-NF
Title:
CONEXAO McMASTER
Number:
EXPERIENCIA McMASTER - LABTOOLS
Date: 15/12/2009
Sheet 2 of 2
Revision: 1_00
File:
MOSAICO HFS
Rua: Galeão Carvalhal, 125
Jd. Bela Vista - 09041400
Santo André, SP - Brasil
55 (11) 4992-8775
www.mosaicohfps.com.br
LabTools
Mosaico Didactic Division

34. Apêndice C – Esquema elétrico completo da placa de experiências EXP03

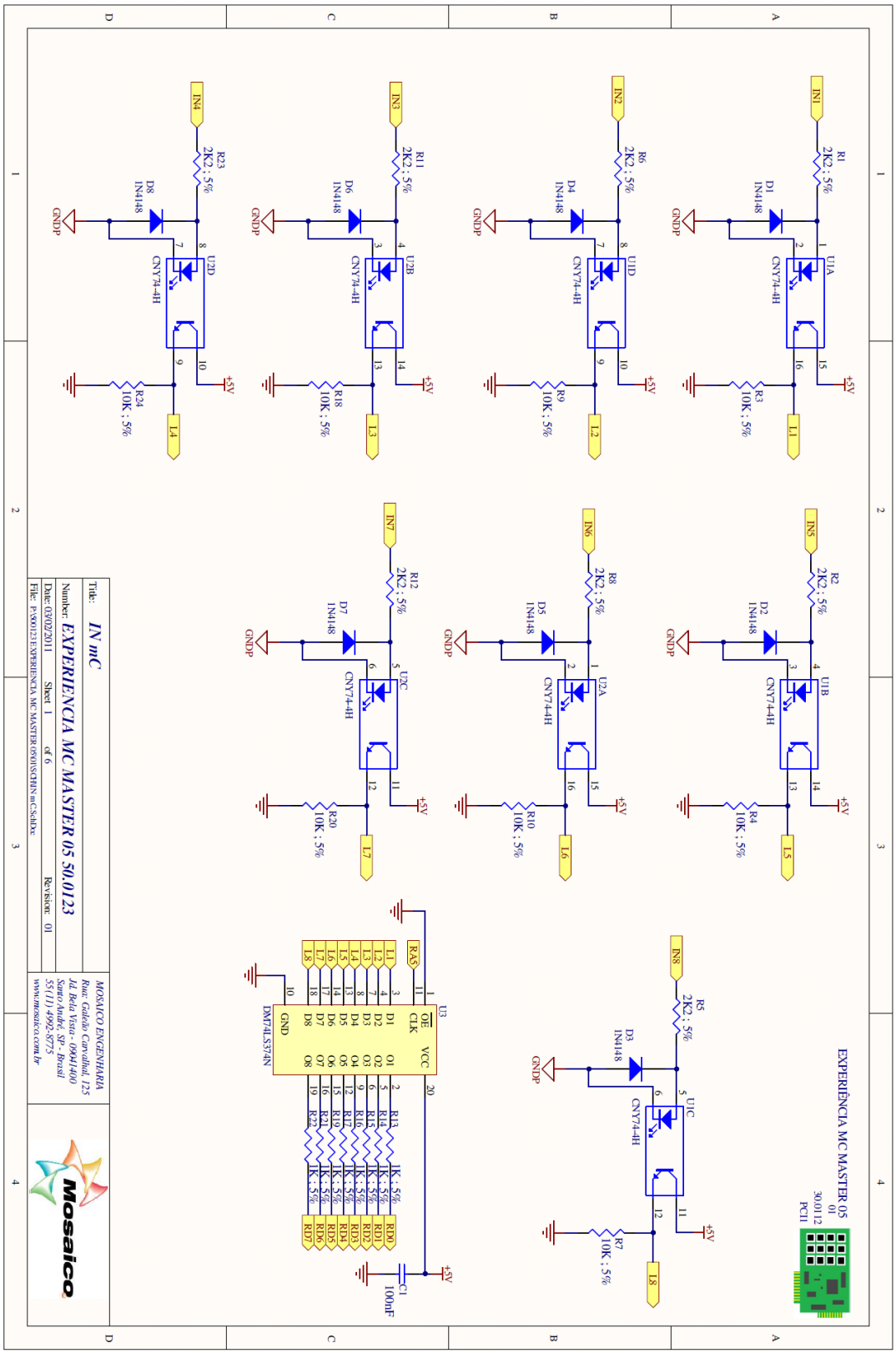


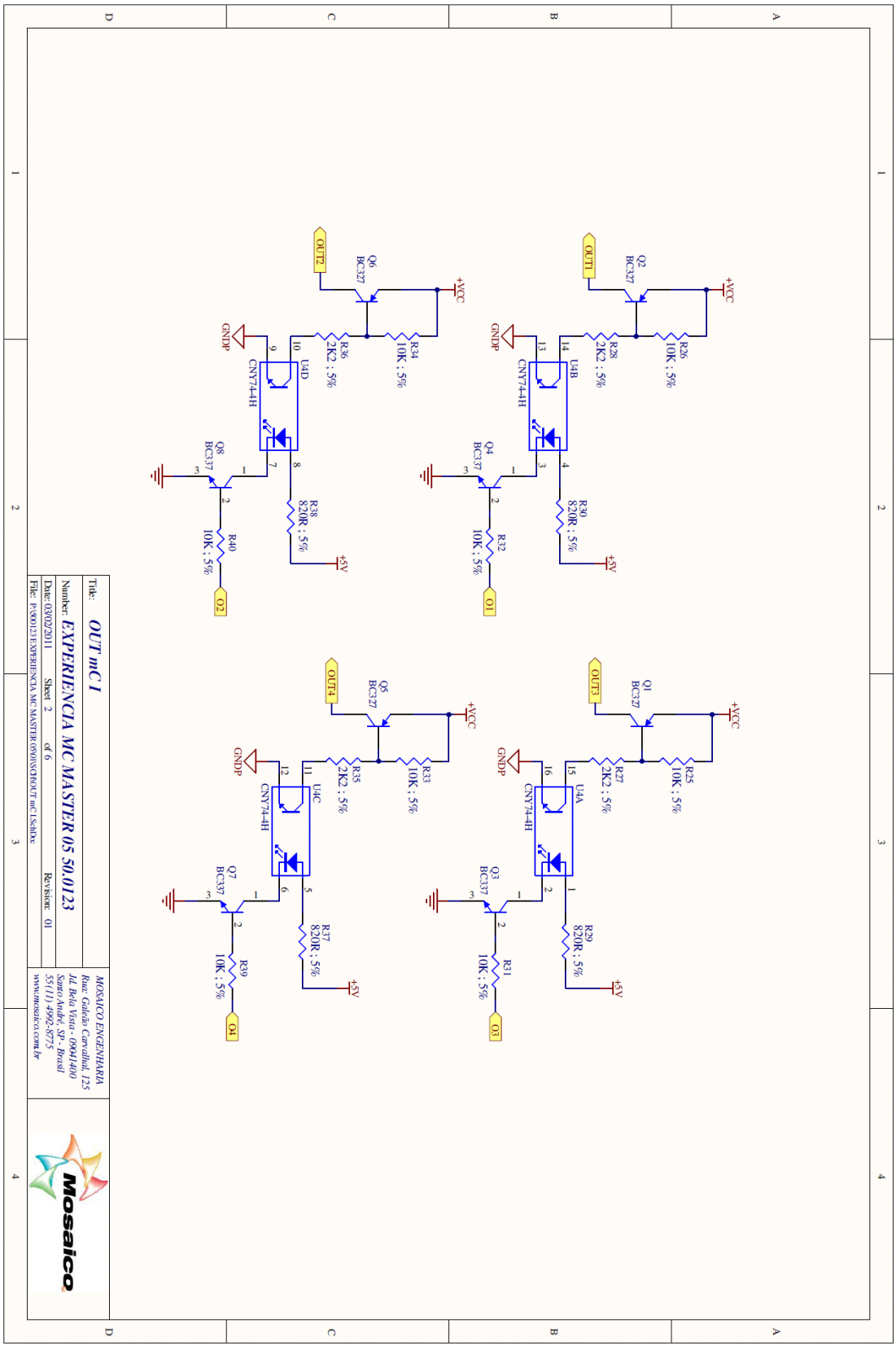

P/ PLACA MCMASTER
CN1
CN2
CN4
GND
+12V
+5V
GNDA
10K ; 5%
SW-DIP8
S1
CN3
KRE
VS
AN0
AN1
AN2
AN3
4K7 ; 5%
CN5
VDD
D-
D+
GND
USB
67068801 1
CONEXÕES
EXP03 Mc MASTER
MOSAICO HPS
Rua: Galeão Carvalhal, 125
Jd. Bela Vista - 09041400
Santo André, SP - Brasil
55 (11) 4992-8775
5/12/2009
Revision: 1_00

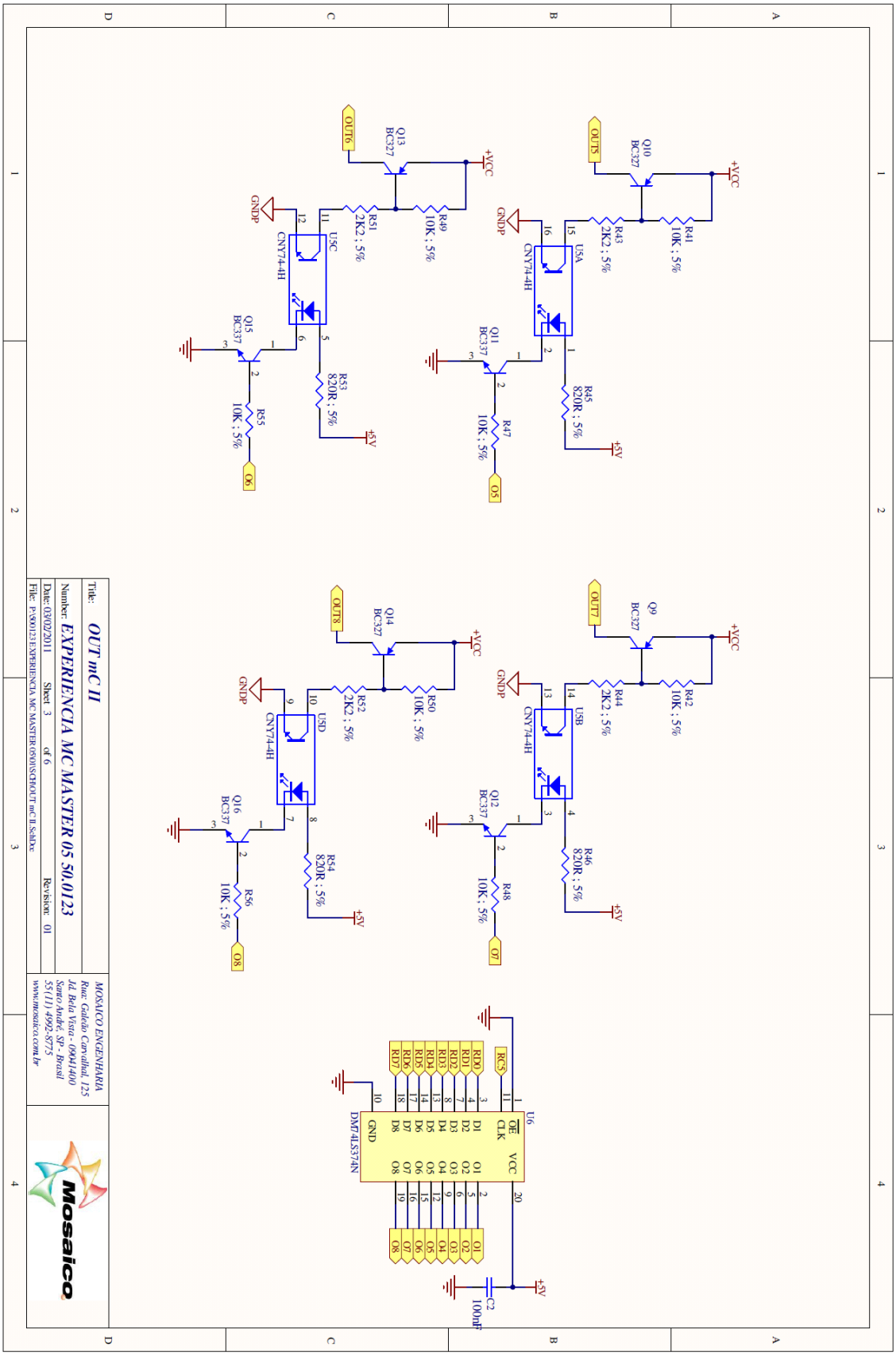
35. Apêndice D – Esquema elétrico completo da placa de experiências EXP04

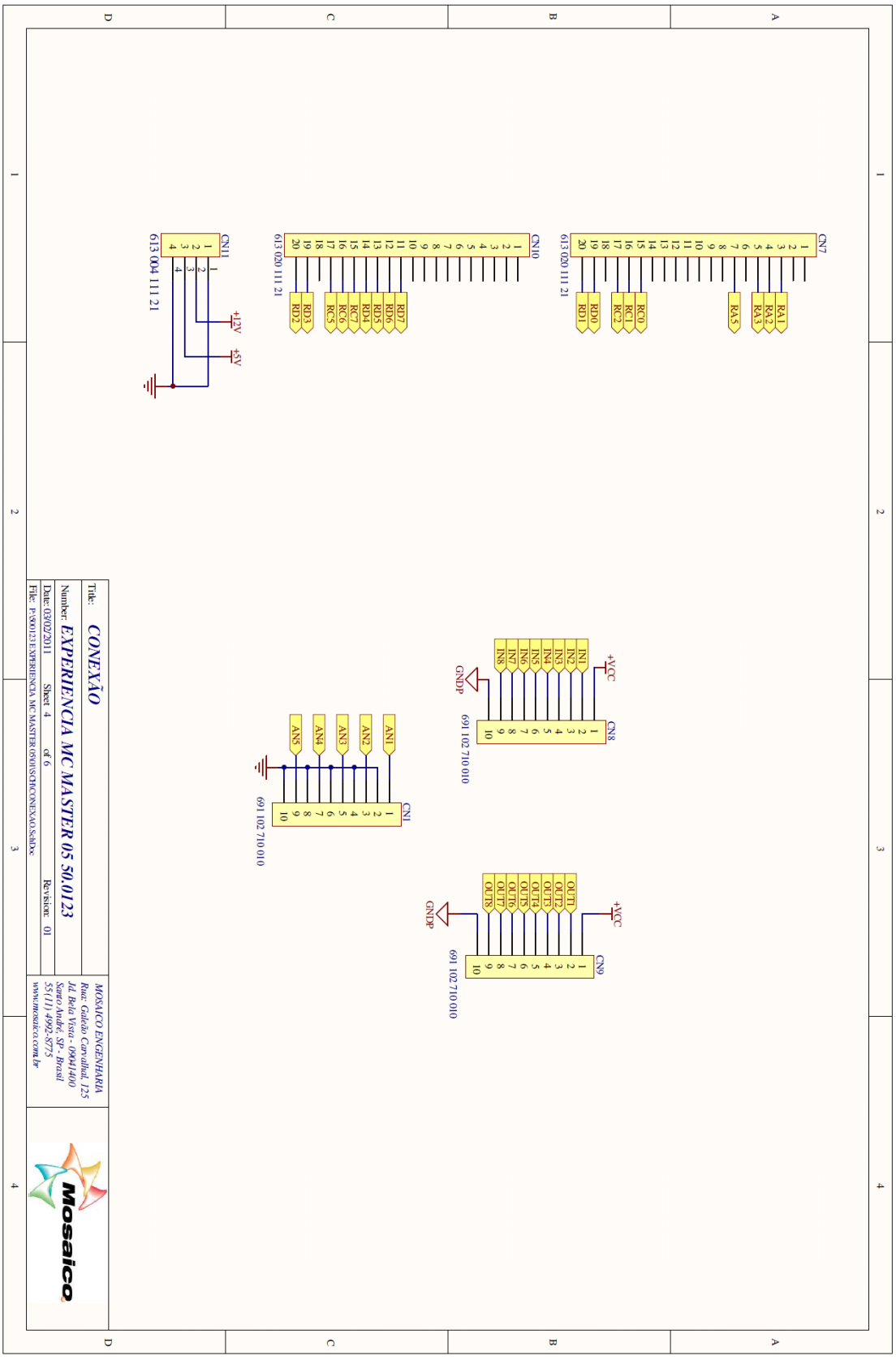


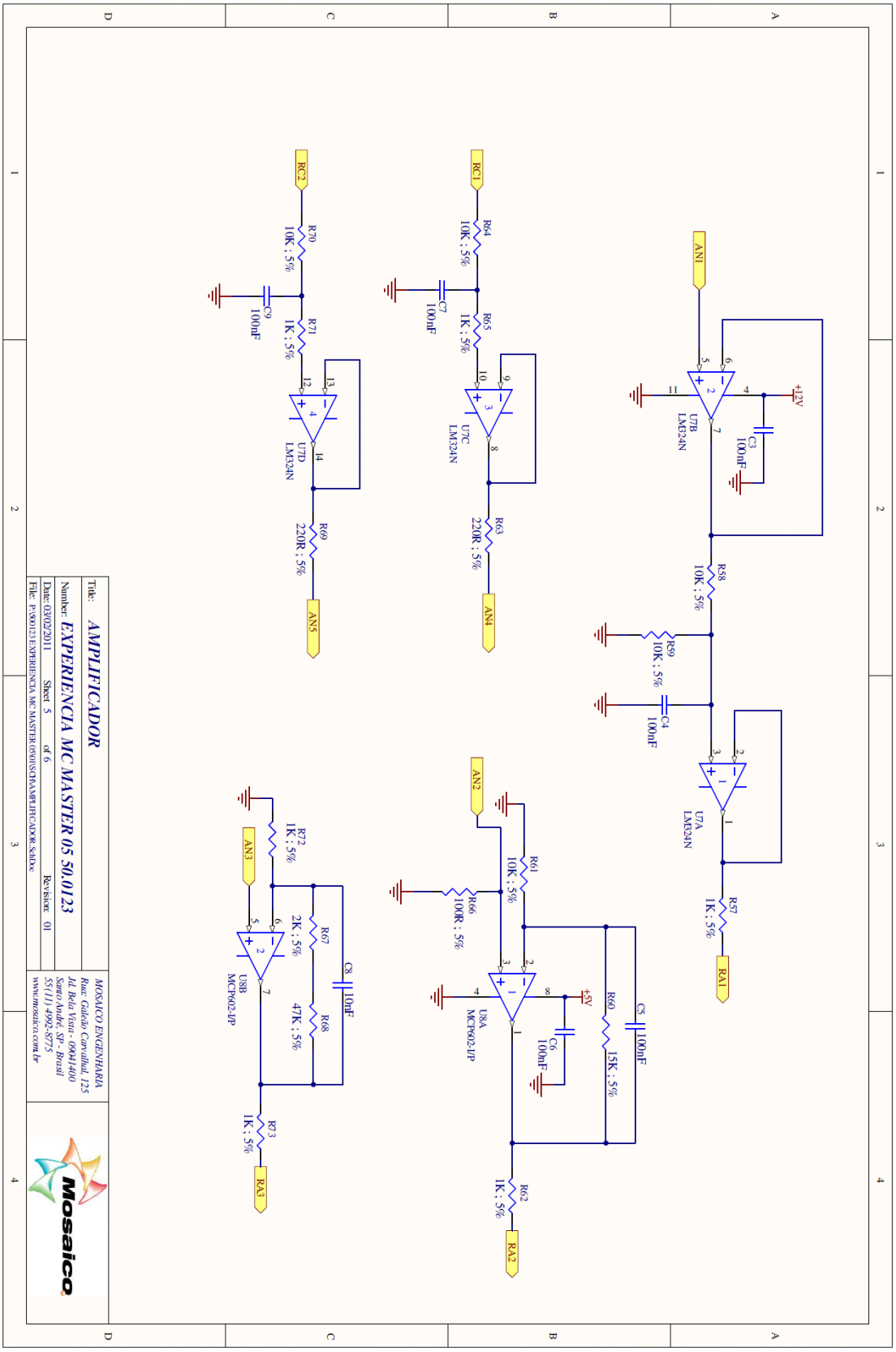
36. Apêndice E – Esquema elétrico completo da placa de experiências EXP05

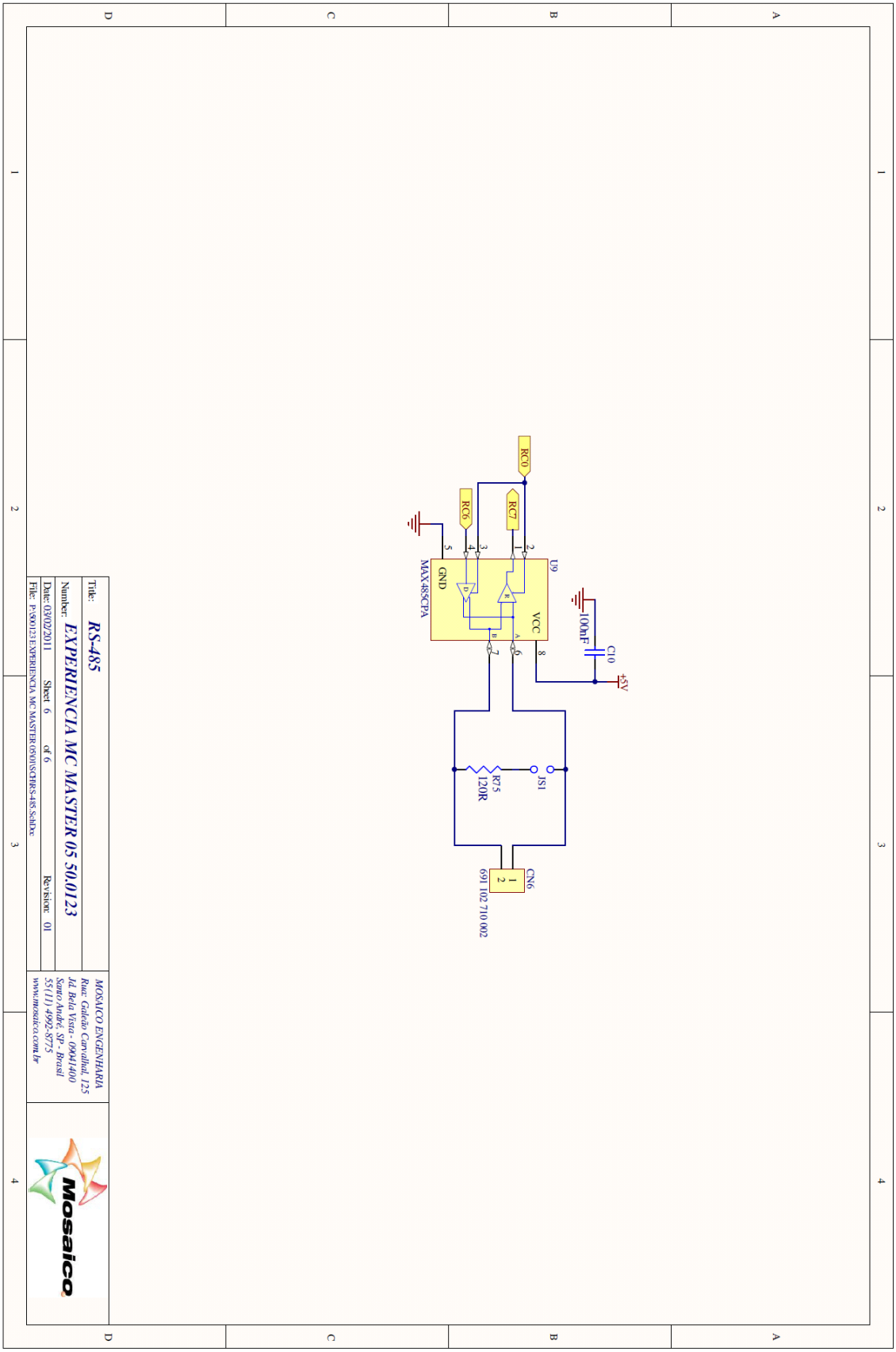



OUT1
OUT2
OUT3
OUT4
+VCC
GNDP
+5V
Q1 BC327
Q2 BC327
Q5 BC327
Q6 BC327
Q3 BC337
Q4 BC337
Q7 BC337
Q8 BC337
R25 10K ; 5%
R26 10K ; 5%
R33 10K ; 5%
R34 10K ; 5%
R27 2K2 ; 5%
R28 2K2 ; 5%
R35 2K2 ; 5%
R36 2K2 ; 5%
R29 820R ; 5%
R30 820R ; 5%
R37 820R ; 5%
R38 820R ; 5%
R31 10K ; 5%
R32 10K ; 5%
R39 10K ; 5%
R40 10K ; 5%
U4A CNY74-4H
U4B CNY74-4H
U4C CNY74-4H
U4D CNY74-4H
O1
O2
O3
O4
Title: OUT mC 1
Number: EXPERIENCIA MC MASTER 05 50.0123
Date: 03/02/2011 Sheet 2 of 6 Revision: 01
File: F:\50123 EXPERIENCIA MC MASTER 05\05SCH\OUT mC 1.SchDoc
MOSAICO ENGENHARIA
Rua: Galeão Carvalhal, 125
Jd. Bela Vista - 09041400
Santo André, SP - Brasil
55 (11) 4992-8775
www.mosaico.com.br

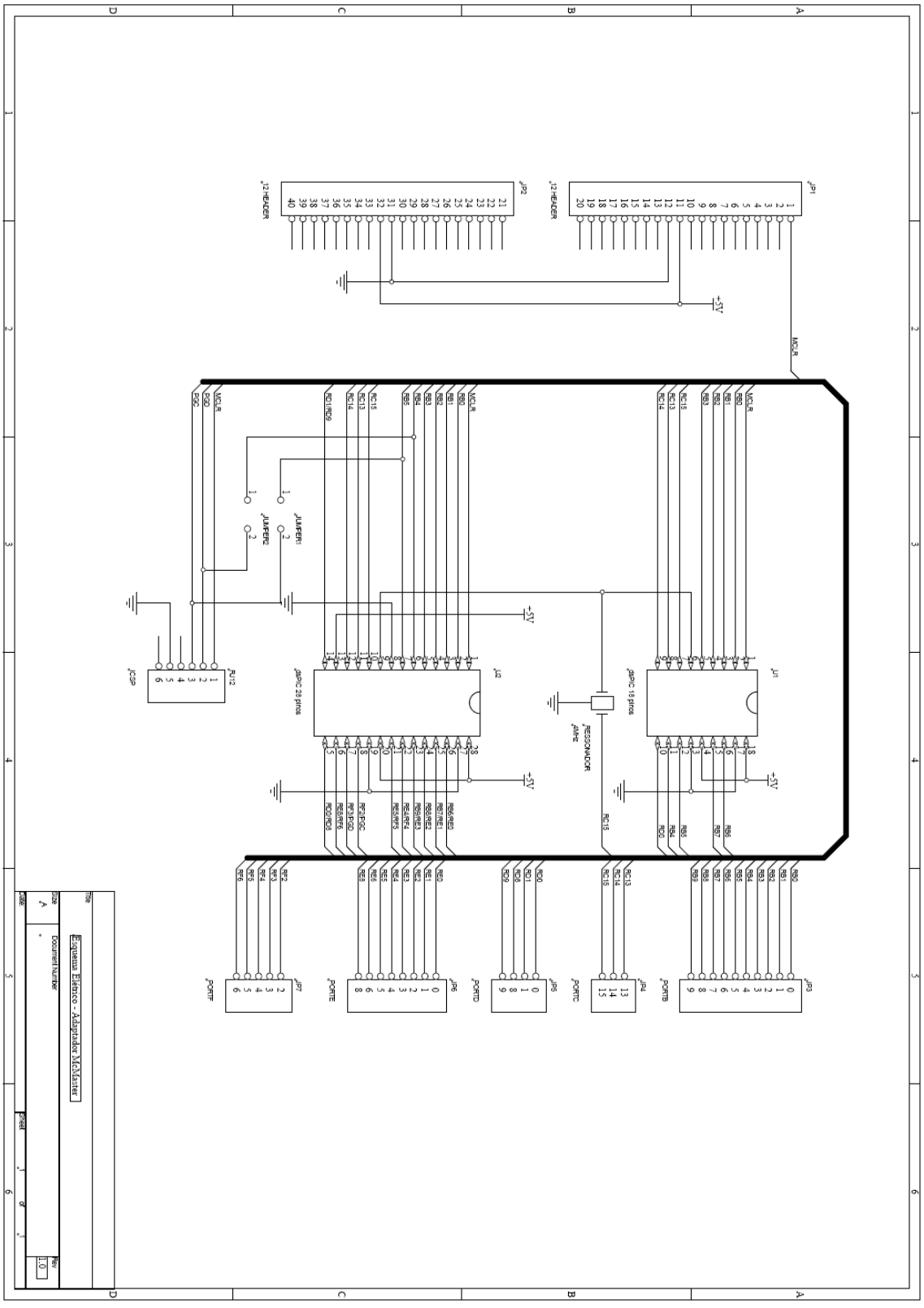

OUT5
OUT6
OUT7
OUT8
Q10 BC327
Q11 BC337
Q12 BC337
Q13 BC327
Q14 BC327
Q15 BC337
Q16 BC337
Q9 BC327
+VCC
GNDP
U5A CNY74-4H
U5B CNY74-4H
U5C CNY74-4H
U5D CNY74-4H
R41 10K ; 5%
R42 10K ; 5%
R43 2K2 ; 5%
R44 2K2 ; 5%
R45 820R ; 5%
R46 820R ; 5%
R47 10K ; 5%
R48 10K ; 5%
R49 10K ; 5%
R50 10K ; 5%
R51 2K2 ; 5%
R52 2K2 ; 5%
R53 820R ; 5%
R54 820R ; 5%
R55 10K ; 5%
R56 10K ; 5%
+5V
O5
O6
O7
O8
U6 DM74LS374N
RC5
RD0
RD1
RD2
RD3
RD4
RD5
RD6
RD7
C2 100nF
Title: OUT mC II
Number: EXPERIENCIA MC MASTER 05 50.0123
Date: 03/02/2011 Sheet 3 of 6
Revision: 01
File: P:\50\0123\EXPERIENCIA MC MASTER 05\05\SCH\OUT mC II.SchDoc
MOSAICO ENGENHARIA
Rua: Galeão Carvalhal, 125
Jd. Bela Vista - 09041400
Santo André, SP - Brasil
55(11) 4992-8775
www.mosaico.com.br


CN7
613 020 111 21
RA5
RA1
RA2
RA3
RC0
RC1
RC2
RD0
RD1
CN10
613 020 111 21
RD7
RD6
RD5
RD4
RC7
RC6
RC5
RD3
RD2
CN11
613 004 111 21
+12V
+5V
CN8
691 102 710 010
IN1
IN2
IN3
IN4
IN5
IN6
IN7
IN8
+VCC
GNDP
CN1
691 102 710 010
AN1
AN2
AN3
AN4
AN5
CN9
691 102 710 010
OUT1
OUT2
OUT3
OUT4
OUT5
OUT6
OUT7
OUT8
+VCC
GNDP
Title: CONEXÃO
Number: EXPERIENCIA MC MASTER 05 50.0123
Date: 03/02/2011 Sheet 4 of 6 Revision: 01
File: P:\50\123 EXPERIENCIA MC MASTER\09\05\MCONEXAO.SchDoc
MOSAICO ENGENHARIA
Rua: Galeão Carvalhal, 125
Jd. Bela Vista - 09041400
Santo André, SP - Brasil
55 (11) 4992-8775
www.mosaico.com.br
Mosaico


AN1
+12V
U7B
LM324N
C3
100nF
R58
10K ; 5%
R59
10K ; 5%
C4
100nF
U7A
LM324N
R57
1K ; 5%
RA1
RC1
R64
10K ; 5%
C7
100nF
R65
1K ; 5%
U7C
LM324N
R63
220R ; 5%
AN4
RC2
R70
10K ; 5%
C9
100nF
R71
1K ; 5%
U7D
LM324N
R69
220R ; 5%
AN5
AN2
R61
10K ; 5%
R66
100R ; 5%
U8A
MCP602-I/P
+5V
C6
100nF
R60
15K ; 5%
C5
100nF
R62
1K ; 5%
RA2
AN3
R72
1K ; 5%
R67
2K ; 5%
C8
10nF
R68
47K ; 5%
U8B
MCP602-I/P
R73
1K ; 5%
RA3
Title: AMPLIFICADOR
Number: EXPERIÊNCIA MC MASTER 05 50.0123
Date: 03/02/2011 Sheet 5 of 6 Revision: 01
File: P:\50\0123 EXPERIÊNCIA MC MASTER 05\05\SCHA\AMPLIFIC\AMPLIFIC.SchDoc
MOSAICO ENGENHARIA
Rua: Galeão Carvalhal, 125
Jd. Bela Vista - 09941400
Santo André, SP - Brasil
55 (11) 4992-8775
www.mosaico.com.br
Mosaico


RC0
RC7
RC6
U9
MAX485CPA
GND
VCC
100nF
C10
+5V
R75
120R
JS1
CN6
691 102 710 002
Title: RS-485
Number: EXPERIENCIA MC MASTER 05 50.0123
Date: 03/02/2011 Sheet 6 of 6 Revision: 01
File: P:\50\0123\EXPERIENCIA MC MASTER\05\05\SCH\RS-485.SchDoc
MOSAICO ENGENHARIA
Rua: Galeão Carvalhal, 125
Jd. Bela Vista - 09041400
Santo André, SP - Brasil
55 (11) 4992-8775
www.mosaico.com.br

37. Apêndice D – Esquema elétrico completo da placa Adaptadora McMaster



37.1.1.1.1.1.1.Pinagem Adaptador McMaster

DsPICs 28 pinos

Pino	Função				Função especial
	30F2012	30F3013	30F3010	30F4012	
1	MCLR	MCLR	MCLR	MCLR	MCLR
2	RB0	RB0	RB0	RB0	
3	RB1	RB1	RB1	RB1	
4	RB2	RB2	RB2	RB2	
5	RB3	RB3	RB3	RB3	
6	RB4	RB4	RB4	RB4	
7	RB5	RB5	RB5	RB5	
8	Vss	Vss	Vss	Vss	GND
9	OSC1	OSC1	OSC1	OSC1	OSC1
10	RC15	RC15	RC15	RC15	OSC2
11	RC13	RC13	RC13	RC13	
12	RC14	RC14	RC14	RC14	
13	Vdd	Vdd	Vdd	Vdd	VCC
14	RD9	RD9	RD1	RD1	
15	RD8	RD8	RD0	RD0	

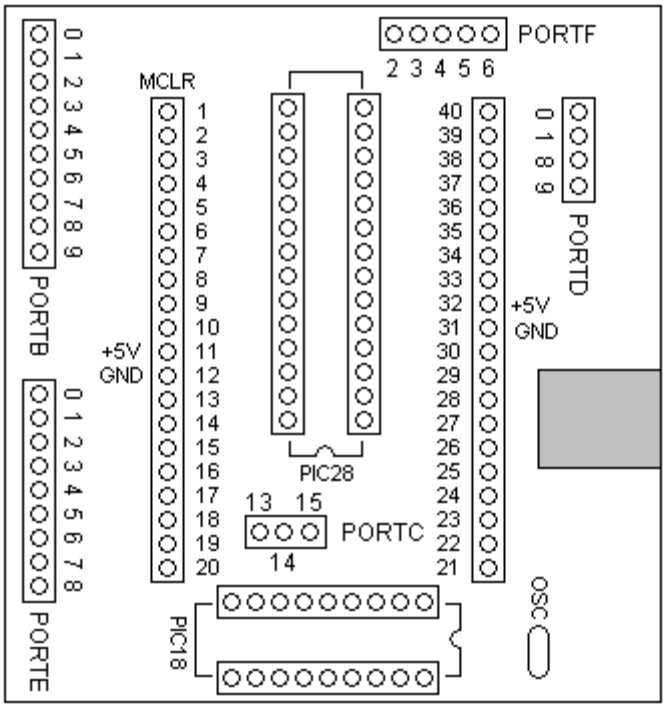
16	RF6	RF6	RE8	RE8	
17	RF3	RF3	RF3	RF3	PGD
18	RF2	RF2	RF2	RF2	PGC
19	Vss	Vss	Vss	Vss	GND
20	Vdd	Vdd	Vdd	Vdd	VCC
21	RF5	RF5	RE5	RE5	
22	RF4	RF4	RE4	RE4	
23	RB9	RB9	RE3	RE3	
24	RB8	RB8	RE2	RE2	
25	RB7	RB7	RE1	RE1	
26	RB6	RB6	RE0	RE0	
27	AVss	AVss	AVss	AVss	GND
28	AVdd	AVdd	AVdd	AVdd	VCC

DsPICs 18 pinos

Pino	**Função**		**Função**
	30F2011	**30F3012**	**especial**
1	MCLR	MCLR	MCLR
2	RB0	RB0	
3	RB1	RB1	

4	RB2	RB2	
5	RB3	RB3	
6	OSC1	OSC1	OSC1
7	RC15	RC15	OSC2
8	RC13	RC13	
9	RC14	RC14	
10	RD0	RD0	
11	RB4	RB4	PGD
12	RB5	RB5	PGC
13	Vss	Vss	GND
14	Vdd	Vdd	VCC
15	RB7	RB7	
16	RB6	RB6	
17	AVss	AVss	GND
18	AVdd	AVdd	VCC

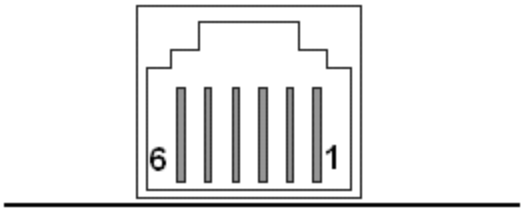
Sugestão do adaptador:

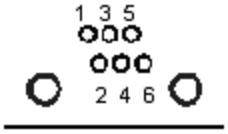


Pinagem do RJ12:

Conector RJ12	
Pino	**Função**
6	Não usado
5	GND
4	Vdd
3	PGC
2	PGD
1	MCLR

Vista do componente:





Frontal

Lado da solda

38. Certicado de Garantia

“PARABÉNS; VOCÊ ACABA DE ADQUIRIR O KIT McMASTER PARA MICROCONTROLADORES PIC DA MOSAICO”

1. Tempo de Garantia

A Mosaico garante contra defeitos de fabricação durante 4 meses para mão de obra de conserto.

O prazo de garantia começa a ser contado a partir da emissão do pedido de venda.

2. Condições de Garantia

Durante o prazo coberto pela garantia, à Mosaico fará o reparo do defeito apresentado, ou substituirá o produto, se isso for necessário.

Os produtos deverão ser encaminhados à Mosaico, devidamente embalados por conta e risco do comprador, e acompanhados deste Certificado de Garantia “sem emendas ou rasuras” e da respectiva Nota Fiscal de aquisição.

O atendimento para reparos dos defeitos nos produtos cobertos por este Certificado de Garantia será feito somente na Mosaico, ficando, portanto, excluído o atendimento domiciliar.

3. Exclusões de Garantia

Estão excluídos da garantia os defeitos provenientes de:

Alterações do produto ou dos equipamentos.

Utilização incorreta do produto ou dos equipamentos.

Queda, raio, incêndio ou descarga elétrica.

Manutenção efetuada por pessoal não credenciado pela Mosaico.

Obs.: Todas as características de funcionamento dos produtos Mosaico estão em seus respectivos manuais.

4. Limitação de Responsabilidade

A presente garantia limita-se apenas ao reparo do defeito apresentado, a substituição do produto ou equipamento defeituoso.Nenhuma outra garantia, implícita

ou explícita, é dada ao comprador.

A Mosaico não se responsabiliza por qualquer dano, perda, inconveniência ou prejuízo direto ou indireto que possa advir de uso ou inabilidade de se usarem os produtos cobertos por esta garantia.

A Mosaico estabelece o prazo de 30 dias (a ser contado a partir da data da nota Fiscal de Venda) para que seja reclamado qualquer eventual falta de componentes.

Importante: Todas as despesas de frete e seguro são de responsabilidade do usuário, ou seja, em caso de necessidade o Cliente é responsável pelo encaminhamento do equipamento até a Mosaico.